Service Level Agreement Specification for IoT Application Workflow Activity Deployment, Configuration and Monitoring

Awatif Alqahtani

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

Newcastle University December 2020

I would like to dedicate this thesis to my loving grandma, parents, mother-in-law, husband and my children

Declaration

I declare that this thesis is my own work unless otherwise stated. No part of this thesis has previously been submitted for a degree or any other qualification at Newcastle University or any other institution. This dissertation contains approximately 50,000 words, excluding appendices, bibliography, footnotes, tables and equations, and has approximately 50 figures.

Awatif Alqahtani
December 2020

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors: Dr Ellis Solaiman, Prof Rajiv Ranjan and Prof Aad van Moorsel for giving me the opportunity and support to complete this research and for all their valuable comments, hints, suggestions and many more insights that inspired this work. I would have been lost without their support and I wish to express my appreciation. I would also like to thank the members of my examining committee Dr. Dhavalkumar Thakker and Dr. Nigel Thomas.

My gratitude is also extended to the School of Computing for giving me the opportunity to undertake this memorable learning experience. I would also like to thank all the people and staff of the school who assisted me at different stages of the research.

I would also like to thank my best friend ever "Sameeha Alahmadi" for her support and for being a lively and entertaining part of the journey and being there whenever I needed her. Additionally, I thank my friends, in particular, "Aisha Blfgeh" and "Shaimaa Bajoudah" for their constant supportive and encouraging words. Special thanks go to Yinhao Li, Adam Booth and Saleh Mohamed for their assistance when I needed it.

I would like to take this opportunity to express my greatest debt to my family, in particular my mother, father, mother-in-law and my brothers, my brothers-in-law, my sisters and my sisters-in-law. They never stopped encouraging me to finish this thesis. Finally, heartfelt thanks go to my husband, "Mohammed Alqahtani", and my children : "Aseel", "Turki" and "Feesl" who have stood behind me and given me huge support during the thesis work; they suffered considerably because of my academic interests. I thank my special daughter "Eqlima" for being part of our family these past two years; she really means a lot to me. I thank God for all that and all the things that have happened to me in my life.

Abstract

Currently, we see the use of the Internet of Things (IoT) within various domains such as healthcare, smart homes, smart cars, smart-x applications, and smart cities. The number of applications based on IoT and cloud computing is projected to increase rapidly over the next few years. IoT-based services must meet the guaranteed levels of quality of service (QoS) to match users' expectations. Ensuring QoS through specifying the QoS constraints using service level agreements (SLAs) is crucial. Also because of the potentially highly complex nature of multi-layered IoT applications, lifecycle management (deployment, dynamic reconfiguration, and monitoring) needs to be automated. To achieve this it is essential to be able to specify SLAs in a machine-readable format.

currently available SLA specification languages are unable to accommodate the unique characteristics (interdependency of its multi-layers) of the IoT domain. Therefore, in this research, we propose a grammar for a syntactical structure of an SLA specification for IoT. The grammar is based on a proposed conceptual model that considers the main concepts that can be used to express the requirements for most common hardware and software components of an IoT application on an end-to-end basis. We follow the Goal Question Metric (GQM) approach to evaluate the generality and expressiveness of the proposed grammar by reviewing its concepts and their predefined lists of vocabularies against two use-cases with a number of participants whose research interests are mainly related to IoT. The results of the analysis show that the proposed grammar achieved 91.70% of its generality goal and 93.43% of its expressiveness goal.

To enhance the process of specifying SLA terms, We then developed a toolkit for creating SLA specifications for IoT applications. The toolkit is used to simplify the process of capturing the requirements of IoT applications. We demonstrate the effectiveness of the toolkit using a remote health monitoring service (RHMS)

use-case as well as applying a user experience measure to evaluate the tool by applying a questionnaire-oriented approach. We discussed the applicability of our tool by including it as a core component of two different applications: 1) a context-aware recommender system for IoT configuration across layers; and 2) a tool for automatically translating an SLA from JSON to a smart contract, deploying it on different peer nodes that represent the contractual parties. The smart contract is able to monitor the created SLA using Blockchain technology. These two applications are utilized within our proposed SLA management framework for IoT.

Furthermore, we propose a greedy heuristic algorithm to decentralize workflow activities of an IoT application across Edge and Cloud resources to enhance response time, cost, energy consumption and network usage. We evaluated the efficiency of our proposed approach using iFogSim simulator. The performance analysis shows that the proposed algorithm minimized cost, execution time, networking, and Cloud energy consumption compared to Cloud-only and edge-ward placement approaches.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The Internet of Things (IoT) is a new computing paradigm in which uniquely addressable objects such as radio-frequency identification (RFID) tags, sensors, actuators and mobile phones become part of the Internet environment [160, 399]. This paradigm opens the door to new innovations that will build a novel type of interaction among things and humans. It enables the realisation of smart cities, infrastructures and services that can enhance quality of life and the utilisation of resources [96]. It is estimated that the number of connected smart objects will reach 212 billion by the end of 2020 [168, 166]. Such a large number of connected smart objects will generate huge volumes of data, which need to be analysed and stored [392].

Storing and processing such large volumes of data is not a trivial task; thus, utilising the flexibility and capabilities offered by Cloud computing is essential [530]. Cloud computing offers a pool of configurable resources (hardware/software) that are available on-demand [135], allowing users to submit jobs to service providers on the basis of pay-per-use. While the IoT provides smart devices with the ability to sense and generate large amounts of data at different data speeds, Cloud computing offers advanced technologies for ingesting, analysing and storing data [108]. Consequently, the number of applications based on IoT and Cloud computing will increase rapidly over the next few years.

The IoT has traditionally delegated most of its workflow activities (e.g., computing, filtering, storing) to Cloud computing as the main infrastructure. However, researchers have argued that for many reasons, outsourcing all IoT workflow

activities to the Cloud is not efficient [372]. IoT delay-sensitive applications (e.g., a remote health monitoring service (RHMS)) cannot depend on a centralised data centre that is situated far away, thereby affecting the end-to-end response time. Another reason for inefficiency is that most data generated by the IoT might not be useful; therefore, it is better to discard data generated at or close to the data sources than to waste resources by transferring all of the generated data to the Cloud.

Edge computing is a computing paradigm that aims to push intelligence (computation) near data sources in order to improve performance. Thus, the emerging Edge computing paradigm is essential because it allows the computation capabilities of Edge resources to be utilised, especially given that the computing capacity of Edge resources is continuously increasing. Moreover, the emergence of Edge computing allows most of the required computations to be performed near the data sources whenever possible, which reduces unnecessary network delays and network usage [372, 284].

Gascon and Asin [47] predicted that in the near future, there will be approximately 54 types of IoT-based applications addressing different domain-specific problems, including the domains of security and emergency, smart environment, smart cities, smart metering, smart water, smart animal farming, smart agriculture, industrial control, retail, logistics, domestic and home automation, and eHealth [96]. Users' expectations of the services provided through the IoT revolution are no different from those of most traditional computer- and Internet-based services in that the services must be delivered within the guaranteed Quality of Service (QoS) level. QoS is an indicator that describes non-functional characteristics such as response time and throughput. QoS requirements can be expressed as Service Level Objectives (SLOs). An SLO is an expression associating each QoS requirement with the target level it is expected to achieve [471] within a Service Level Agreement (SLA). An SLA is a contract between a service provider and a service consumer (with the possibility of also involving signatory parties/third parties) that lists the agreed-upon terms of the QoS requirements [165]. In addition to guaranteeing the QoSs, an SLA indicates the actions required if this guarantee is violated [432, 262].

SLAs have been used in many IT-related fields and platforms over many years [77]. For example, the Web Service Level Agreement (WSLA) was introduced in 2003. The WSLA is a framework composed of SLA specifications and a number of SLA monitoring tools for web services [228]. The WSLA aims to allow consumers and service providers to explicitly define and determine the measurement of SLA parameters. Furthermore, WS-Agreement [38] is a Web Service Agreement which defines the specification of the web service agreement as a domain-specific language. Thus, service providers can utilise the WS-Agreement as a protocol to advertise their resource capabilities and create agreements with end users.

In Cloud computing, there are a number of works related to SLAs. For example, an expert group of the European Commission published a report titled "Cloud Computing Service Level Agreements – Exploitation of Research Results" [252]. This report surveys European and national projects, presents research outcomes and discusses the outcomes from an SLA lifecycle perspective. Based on the projects discussed in [252], there are a number of works that have contributed to SLA specifications, such as the blueprint concept in the 4CaaSt project [376]. A blueprint is a descriptive document that expresses the service dependencies across and within Cloud layers. Another project that has contributed to SLA specifications is SLA@SOI [463]. SLA@SOI is a framework that addresses multi-level, multi-provider SLA lifecycle management within a service-oriented architecture and Cloud computing. Within the SLA specification phase, SLA@SOI provides a description of a service called SLA Template (SLA(T)). SLA(T) is an abstract syntax template that uses notations for describing the functional and non-functional characteristics of a service; these notations can be modelled later using Extensible Markup Language (XML), Web Ontology Language (OWL), or any other format.

Traditional SLAs that focus on availability and reliability are not enough for IoT applications due to the need for strict SLA guarantees (of functions such as accuracy and the speed of the detection of the event of interest) [489]. Furthermore, having an individual SLA management mechanism for each layer of the IoT is inadequate because of the huge dependency across layers [28]. Moreover, the majority of service providers available, such as Cloud providers, offer a descriptive summary of the terms and conditions of their services. This poses certain

disadvantages, such as uncertainty and no facility to automate the searching for services or the negotiation of contractual terms [472]. Therefore, providing a machine-readable [1] SLA specification is crucial.

The importance of providing machine-readable SLA specification is also highlighted by the need for formal guarantees that the services offered comply with the terms negotiated, as Cloud users may, for example, outsource their core business functions to the Cloud [472]. Many languages have been proposed for defining machine-readable SLAs and to simplify their assessment and negotiation [472]. Nevertheless, available SLA frameworks vary between being too specific or too generic [165]. Therefore, we argue that these languages cannot cope with the IoT's distinctive features, such as multi-layer multi-provider nature of IoT agreements and deployment models. In IoT applications, there is a need to aggregate QoS requirements from the perspectives of the cooperated layers such as Cloud, network, and sensing layers. The main purpose of considering QoS across layers is to deliver the promised IoT functionalities that match consumers' expectations at the application level, as agreed upon within the SLA.

1.1 Motivation and research problem

To shed light on the complexity of IoT applications needing end-to-end SLA specification, we considered three of the best-known computing paradigms: Cluster, Grid, and Cloud, illustrating the discrepancies between them and the IoT. Cluster computing, for example, is a type of computing that makes several nodes run as a single entity [226, 12]. All the nodes on the system can simultaneously run the same application. It is a system where computers (processing elements) work together to accomplish tasks. In Grid computing, resource segregation (separation) from multiple sites is used to solve a problem that cannot be solved by using single computer processing [226, 12]. Users have no or little knowledge about where these resources are placed or what the underpinning infrastructures, operating systems, hardware or software are [213, 12]. Unlike Clusters that have to be onsite, Grids are distributed across the globe; that is, they use Internet power to link resources together irrespective of their geographical location. This

[1] Machine readable: "a data format that can be automatically read and processed by a computer, such as CSV, JSON, XML, etc. Machine-readable data must be structured data" [494]

moves the emphasis in Cluster computing from performance to resource sharing, eliminating the need for Single System Image (SSI) as long Grid machines are heterogeneous and geographically dispersed [12].

Cloud computing is a model that makes it possible to access a common pool of configurable computing resources (i.e., networks, servers, storage, apps and services) on-demand. The computing resources are delivered quickly and released with minimal management effort or interaction from the service provider [118, 12]. Thus, unlike Cluster or Grid computing, where the focus is on processing resources to solve the problem, Cloud computing is about delivering on-demand services [12].

The IoT is a system of devices that are equipped with unique identification (UID) and the possibility of the network sharing of data without the need for human-to-computer interaction [493]. Owing to the integration of various innovations – real-time computing, artificial intelligence, sensors and embedded devices – the IoT has developed [493]. Embedded systems (including home and building automation), wireless sensor networks, control systems, and other technologies all contribute in IoT.

In Cluster computing, Grid computing or Cloud computing, the emphasis is on computation power to solve problems and the provisioning of services on-demand. On the other hand, the emphasis in the IoT is on combining performance, the provision of services on-demand, and the distribution of computing power in order to allow real-time processing while respecting SLA constraints at the application level (e.g. the response time of an application is less than 5 milliseconds). Therefore, we argue that it is essential to propose an SLA for the IoT on an end-to-end basis. By ‘end-to-end basis’ we mean considering the constraints of the QoS and configuration requirements for all the involved components (services and infrastructure resources) that are part of an IoT ecosystem. Therefore, we introduce a workflow activity term within our proposed SLA specification (which is introduced in Chapter 3) to allow us to capture requirements across layers and within the layer itself.

One of the factors that raises the importance of specifying the SLA on an end-to-end basis is the fact that the IoT can be delivered by many providers and each provider might be a consumer as well. As a result, SLAs in the IoT have a strong dependency relationship with each one of the whole system's components, regardless of whether this component is hardware, software or a human being. This means that a violation of one or more constraints by one or more actor(s) affects the adherence to the quality of service at application level. To illustrate the concept, consider a Remote Healthcare Monitoring Service (RHMS) where patients wear sensors and accelerometers to measure their heart rate and sugar levels, reminding them when it is time to take medications and detecting abnormal activity such as falling down. Patients can register in a remote healthcare monitoring service and pay for this service. They can then be sent to the hospital as an emergency case and their caregivers and doctors will be alerted whenever their health is/might be in a critical situation. Subscribed patients are looking for a service that can satisfy the following high-level requirement: detecting abnormal activity such as falling down, within x milliseconds, and contacting the ambulance, caregivers and doctors within y minutes.

From the above scenario, adherence to SLAs in the IoT is a critical process and complex since it can be seen that in order to achieve the SLA at the application level (e.g., end-to-end response time), many nested-dependent QoS should be considered. For example, as patients need to receive the required aid based on their health status within Y minutes, that means that the aggregation of the required time for detecting/transferring/analysing /alerting should be within the time constraints, i.e. less than or equal to Y minutes. That requires high-quality sensors with minimum event detection delay, available networks with low latency, and a notification service with low response time to deliver the desired value of the application. In Figure 1.1, we can see that in order to respond within the expected time constraints, any delay within the dotted ovals or arrows are counted and can affect the quality level of the delivered response. For example, if there was a delay in filtering data activity, it would lead to a late response at the front-end which then might exceed what the consumer was expecting. Consequently, if the response time was behind consumer expectations, this might lead to catastrophic results, especially if it were a matter of life and death.

Fig. 1.1 An example of dependency between workflow activity

Although large efforts have been made by big companies in the field such as Amazon and Microsoft, which offer platforms that enable users/applications to connect their devices and benefit from the available cloud services, there are still limitations on the guaranteed quality of services. A number of works consider SLA specifications for all the Cloud tiers or just one of the Cloud tiers. For example, a Cloud Service Level Agreement (CSLA) [245] is a specification language developed specifically for the Cloud domain. Another example is Service Level Agreement Language for Cloud Computing (SLAC), which is defined by [476]. However, the focus of SLAC is only related to Infrastructure as a Service (IaaS) [476]. Furthermore, while an effort has been made to develop an SLA specification for Networking, as far as we know, no works address SLA specifications in such a way that they consider SLAs on an end-to-end basis by considering the IoT layers (described in Chapter 2): the IoT devices layer, Edge layer, Cloud layer and application layer. Therefore, the performance of an RHMS relies not only on the correctness of the provided functionalities but also on the quality of the offered services across the Edge and/or Cloud computing environments. Therefore, SLAs undoubtedly need to consider requirements across all layers of the Edge and/or Cloud environment – for example, at what rate data should be collected, transferred, and ingested and how fast and accurate the analysis should be.

In IoT applications, there is a need for strict SLA guarantees [489]. Thus, within an end-to-end SLA, it is necessary to express constraints/policies that determine which data can be processed within the Edge data centres as well as which data need to be exported to be processed/analysed in Cloud data centres under certain constraints. Additionally, specifying the contractual terms of an SLA on an end-to-end basis is important to assure consumers that their QoS requirements will be observed across layers. Therefore, for such applications and others, ensuring that consumer requirements are accurately and unambiguously specified within SLAs is crucial. Accurately specified SLAs are contracts that can form the basis of a strategy to regulate and automate transactions and activities

between interacting parties (service providers and consumers).

When specifying SLA terms on an end-to-end basis within a formal syntax language, standardising the vocabularies used to describe the offered/requested services is crucial. With the multi-layered nature of IoT applications, it is possible to have more than one provider. Having multiple providers is a serious issue that requires the terminologies used within the SLA to be standardised in order to avoid ambiguity. For example, within the Cloud environment, there is a lack of standardised vocabularies for expressing SLAs. For example, availability is expressed differently by well-known Cloud providers: Amazon EC2 offers availability as a monthly uptime percentage of 99.95%; Azure offers availability as a monthly connectivity uptime service level of 99.95%; and GoGrid offers a server uptime of 100% and an uptime of the internal network of 100% [24]. Furthermore, within the Edge environment, sampling rate [214] and sampling frequency [280] are used interchangeably to describe the rate at which a sensor sends data. Therefore, standardising the vocabularies used to describe the offered and requested services can play a significant role in minimising the ambiguity between cooperating parties which in turn could provide successful interactions between consumers and providers.

Furthermore providing machine-readable SLA is important not only for SLA management purpose but in service provider's selection. A consumer who wishes to start an SLA must first select a service provider/s. Selecting service provider/s can be a challenging process, especially when considering the multi-layered nature of the IoT. Since IoT applications have a multi-layered architecture, IoT administrators need to consider different categories of providers (e.g., network provider, Cloud provider) and find the best candidate for each category. Most popular Cloud providers (e.g., AWS, MS Azure, Oracle) currently provide descriptive take-it-or-leave-it SLAs for their services. When consumers need to compare these SLAs from different providers to select the most suitable, they must evaluate them manually [495]. IoT applications can potentially be much more complex than Cloud applications, and such a comparison therefore becomes difficult. Therefore, standardising the vocabularies used to describe the QoS of the offered and requested services can be a first step towards enhancing and automating the process of selecting service providers using certain search

criteria.

Not having a machine-readable format [2] has many disadvantages. For example, it creates confusion regarding SLA interpretation and makes automation of the SLA lifecycle[3] infeasible [472]. In addition to the importance of standardising SLA vocabularies and structures, providing SLAs in a machine-readable format is an important step towards automating the process of application deployment, monitoring, and dynamic re-configuration [472, 290]. For example, once an IoT application has been deployed, it is important to continuously monitor the extent to which the application adheres to what has been agreed upon, as well as to reconfigure the application dynamically, on the fly, as needed to avoid/minimise SLA violations [337].

Nevertheless, specifying and managing end-to-end SLA cross-computing environments is not a trivial task, since there are a number of challenges that need to be addressed [28] [392]:

- **Heterogeneity of key QoS metrics across computing environments:** Considering key performance metrics and their variation across computing environments and within their layers is crucial. There are different QoS metrics for each layer, which are not necessarily the same [214]:

  1. application layer (e.g., event detection and decision-making delays).
  2. Cloud layer (e.g., the QoS of big data frameworks such as the throughput of batch processing and the QoS of the infrastructure layer, such as CPU utilisation and memory utilisation).
  3. Edge environment (e.g., gateway throughput and latency).
  4. IoT devices (e.g., precision and data quality).

  Therefore, it is essential to provide a coherent taxonomy that considers various QoS metrics for the involved computing layers.

[2] A machine-readable format means an SLA can be read and processed by a computer such as CSV, JSON, XML, etc., and the data are structured.

[3] The SLA lifecycle consists mainly of discovering the service provider, defining the SLA, establishing the agreement, monitoring the SLA, terminating the SLA, and enforcing penalties if there is a violation; for more details about the SLA lifecycle, refer to 2.1.1

- **Heterogeneity of application requirements:**
  An IoT application has specific requirements according to its purpose and domain. For example, Smart Home applications place a high priority on energy consumption, while environmental prediction applications place a high priority on data accuracy and action response time.

- **Cross-layer dependencies:**
  Some issues related to the dependency nature of IoT applications need to be addressed in order to meet end-to-end SLA cross-computing paradigms. For example, if the consumer needs to process collected data within a time constraint, then the end-to-end execution time is affected not only by the processing time at the Cloud layer, if using a Cloud resource, but also by the time it takes to collect and transfer data.

1.2 Research Aim and Questions

To address the challenges discussed earlier, the high-level aim of this thesis is to "design and develop an SLA specification language for IoT application workflow activity deployment, dynamic re-configuration, and monitoring on end-to-end basis". To address this aim, this thesis addresses the following research questions:

1.2.1 Question 1

How can an end-to-end SLA for an IoT application be specified?
It is well known that SLA specification languages for various application domains do indeed exist; for example, see [3, 77, 165, 316, 347, 399]. However, in their current formats, to the best of our knowledge, there is a lack of consideration for the requirements of all the layers (end-to-end) that cooperate to deliver an IoT application.
Providing an answer to this question implies several sub-tasks:

- Provide/select an IoT reference architecture – a number of research studies regarding IoT architecture exist, implying variety in the proposed/studied IoT architecture. Therefore, selecting a reference architecture is a necessity.

Consequently, we present a reference architecture of IoT ecosystems. Furthermore, we provide background information about SLAs and we present a review of the existing research related to the SLA lifecycle over the last decade. This is presented in Chapter 2.

- Propose An SLA conceptual model- The purpose is to provide the meta-model for the proposed SLA specification. Such a model can capture the main concepts that need to be considered within the SLA and the relationships among them. We then evaluate the proposed conceptual model using a questionnaire-oriented approach. This is presented in Chapter 3.

- Provide a predefined list of vocabularies related to the QoS metric and common configuration parameters for the considered IoT ecosystem components. The conceptual model introduces the related vocabularies to specify QoS and configuration parameters as a step to unify used terminologies. This is presented in Chapter 3.

- Present a new multi-layered context-free grammar to describe the recursive syntactic structure of the SLA specification formally for IoT applications. Then, we evaluate the proposed grammar in Chapter 4.

- Provide a Graphical User Interface (GUI)-based tool to generate an end-to-end SLA in a machine-readable format with comprehensive vocabularies. This increases the SLA's inter-portability between the IoT ecosystem components to deal with the integration of different services provided by different providers. Furthermore, the purpose of this step is to simplify the process of generating the SLA for those who are interested in doing so. Thus, they will not have to worry about the correctness of the syntax of the SLA specification when specifying their requirements. The tool allows providers and consumers to express their capabilities and requirements, respectively, on a fine-grained level of details. We demonstrate the tool employing a use case. Then, we evaluate the usability and generalizability of the tool for capturing the requirements of different use cases. This is presented in Chapter 5.

- an SLA management framework for IoT which is mainly consists of SLA specification, negotiation, monitoring and enforcement phases. Furthermore, we provide a mechanism to automatically convert a machine-readable

SLA for IoT into a smart contract that is capable of automatically monitoring adherence to the specified QoS requirements. The generated SLA-based smart contract is utilized in the SLA monitoring phase of the proposed SLA management framework. Furthermore, we present a proof of concept to show that providing a machine-readable SLA can be utilised in more than one phase of the proposed SLA management framework. This is presented in Chapter 6.

1.2.2 Question 2

Ensuring that the SLA is enforced requires the application of several SLA management policies. Therefore, in our second research question we investigate how decentralising workflow activities across Cloud and Edge layers aids the process of adhering to the SLA, for example by reducing the cost, time, network usage, and power consumption?
Providing an answer to this question implies the following sub-tasks:

- An SLA-aware heuristic algorithm to decentralise workflow activities among Edge and Cloud resources. The Algorithm aims to reduce latency, Cloud energy consumption, Cloud cost, and network usage for IoT applications. This is presented in Chapter 7.

- Evaluate the effectiveness of the proposed algorithm using iFogSim[4].

1.3 Publication

- **A. Alqahtani**, K. Alwasil, N. Ayman, K. Mitra, E. Solaiman and R. Ranjan, " The Integration of Scheduling, Monitoring, and SLA in Cyber Physical Systems," Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things, 2020 [**There are parts of it included partially in CHAPTER 2 and 3**]. (Note: *My contribution is discussing SLAs in Cyber Physical Systems as well as presenting and discussing the challenges associated with the Integration of scheduling, monitoring and SLA in Cyber Physical Systems.*)

[4]iFogSim is an open-source toolkit for modelling and simulating resource management approaches for the IoT, Edge and Fog computing https://github.com/Cloudslab/iFogSim

- **A. Alqahtani**, E. Solaiman, R. Buyya and R. Ranjan "End-to-End QoS Specification and Monitoring in the Internet of Things," Newsletter, IEEE Technical Committee on Cybernetics for Cyber-Physical Systems, Volume 1, Issue 2, August 01, 2016.

- **A. Alqahtani**, Y. Li, P. Patel, E. Solaiman and R. Ranjan, " End-to-End Service Level Agreement Specification for IoT Applications," The International Conference on High Performance Computing & Simulation (HPCS 2018).[**Covered in CHAPTER 4**].

- **A. Alqahtani**, P. Patel, E. Solaiman and R. Ranjan, " Demonstration Abstract: A Toolkit for Specifying Service Level Agreements for IoT applications," The International Conference on High Performance Computing & Simulation (HPCS 2018), IoT especial session. [**Covered in CHAPTER 5**].

- **A. Alqahtani**, P. Patel, E. Solaiman, S. Dustdar and R. Ranjan,"Service Level Agreement Specification for End to End IoT Applications Ecosystems", Software: Practice and Experience - Wiley Online Library *is Accepted as a Journal Paper* [**Covered in CHAPTER 4,5**].

- Y.Li, **A. Alqahtani**, E. Solaiman, C. Perera, P. P. Jayaraman, R. Buyya, G. Morgan, and R. Ranjan, "IoT-CANE: A Unied Knowledge Management System for Data-Centric Internet of Things Application Systems," Journal of Parallel and Distributed Computing (JPDC), https://doi.org/10.1016/j.jpdc.2019.04.016, Elsevier. *is Accepted as a Journal Paper* [**partially referred to in CHAPTER 7**]. (Note: *My contribution is providing explanations related to the knowledge base, which is derived from integrating the SLA specification tool and IoTCANE in the implementation phase.*)

- **A. Alqahtani**, P. Patel, E. Solaiman and R. Ranjan, " SLA-aware Approach for IoT Workflow Activities Placement based on Collaboration between Cloud and Edge," Accepted in Fisrt Workshop on Cyber-Physical Social Systems (CPSS) 2019 [**Covered in CHAPTER 8**].

Chapter 2

Background

Overview

In this chapter, we provide a background to the basic technologies that are related to the research carried out within this thesis. We start by presenting the background information related to Service Level Agreements (SLAs) and the Internet of Things (IoT) in Section 2.1. We conduct a systematic mapping study to collect research that is related to SLAs for the Cloud, Edge and IoT from a technical perspective. The aim is to identify current research topics in SLA, particularly for the IoT. We map around 400 papers from different scientific databases. We identify two main categories that most research work related to SLAs falls into: work related to the SLA lifecycle (Section 2.3.1) and work that focuses on SLA applications (Section 2.3.2).

The results show that around two-thirds of the papers focus on the SLA lifecycle: SLA specification, SLA negotiation, SLA monitoring, SLA enforcement, and SLA management. The remaining papers focus on SLA applications such as SLA-aware resource allocation, scheduling applications or other SLA-related topics. We track growth in SLA research through the last 10 years, and we address some of research gaps that need to be considered in future studies

2.1 Background

The IoT is a field about connecting everyday physical objects and devices (such as washing machines, cars, etc.) to the Internet. These devices could share data

about their surroundings via sensors or they could be remotely controlled by their users via smartphone applications. One popular example of an IoT application is a connected car (e.g., Ola Cabs, Uber) that can be tracked and rented using a smartphone. However, one of the IoT's limitations is its limited computing and storage capacity, which has made it necessary to move storage and processing to powerful resources. Therefore, Cloud computing plays a significant role with its processing and storage capability, specifically with its pay-as-you go model.

"A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualised computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on SLAs established through negotiation between the service provider and consumer"[95]. The largest benefit of Cloud computing is that the resources are shared through a shared infrastructure.

The advancements in Cloud computing and its computational technology have led many big-name companies (e.g., Google, Amazon, IBM, Microsoft) to nurture this popular paradigm as a utility. As a result, Cloud-based services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), etc. have emerged, which has increased the number of applications that benefit from Cloud services.

This kind of integration between the IoT and Cloud computing paradigms allows many sources (sensors, humans, applications) to start generating data and organisations tend to store this data for a long time due to their inexpensive storage and processing capabilities [517]. While the centralised architectures of Cloud computing play a role in creating an effective economy, considering a logical extreme, a full centralisation approach could bring unintended results [284]. López et al. [284] mention four fundamental problems with centralised approaches: first, it is necessary to make a trade-off between releasing personal and sensitive data, using centralised services such as social networks, location services, and privacy. Another fundamental problem is that users of Cloud services delegate control of the applications and systems to the Cloud. A third fundamental problem relates to using Cloud resources and neglecting the fact that new generations of Edge devices are embedded with high computational capacity and the ability to communicate seamlessly. Consequently, sufficient stor-

age space is a misuse of that embedded intelligence power. Another fundamental problem is that Cloud-based centralisation hinders human-centred designs, which limits the link between man and machine. Therefore, moving computations to the Edge under certain conditions will minimise the Cloud-based centralisation issues as well as enhance the utilisation of Edge devices' computational power.

Edge computing's fundamental idea is to bring the computing facilities closer to the data source. Edge computing performs lightweight-computational and analytical operations (e.g., filtering, analysing, detecting, etc.) on the received IoT data to improve the performance, save unnecessary data transfers, accelerate decision making, and carry out automatable actions on physical environments. Edge computing is more secure and private than Cloud computing, as sensitive data can be processed and stored more securely [284]. However, Edge computing resources do not offer the same high throughput resources as those offered by the Cloud. Therefore, cooperation for data processing between the Cloud and the Edge is essential for a trade-off between energy efficiency, cost, latency, and bandwidth [366, 225].

Since this research focuses on SLA specifications for the IoT, priority is given to having an IoT architecture to refer to when specifying requirements of the main elements that should be covered within SLAs.

IoT architecture

The IoT paradigm allows billions or trillions of heterogeneous devices to be connected in a seamless manner. Therefore, it is essential to have a flexible IoT architecture that can meet different application needs. The ever-increasing number of proposed architectures has not converged into a model of reference [14]. However, some projects, such as IoT-A [1] are attempting to design a common architecture based on the analysis needs of both researchers and industry [14]. A number of proposed models consider IoT architectures that consist of three layers, while other works consider IoT architectures that consist of four-layer and five-layer architectures. For example, authors in [288] and [106] propose a four-layer architecture that includes the following:

[1] For more information about IoT-A: https://iotforum.files.wordpress.com/2013/07/iot-a.pdf

- Edge Technology Layer: This is a hardware layer where embedded devices, sensors, and tags are located. This layer is responsible for collecting data from a system or an environment and it also offers information processing and communication support.
- Access Gateway Layer: Data handling, publishing and subscribing services, message routing and communication support are the functionalities offered by this layer.
- Middleware Layer: This layer is responsible for aggregating and filtering received data, performing information discovery, and controlling applications' access to the devices.
- Application layer: This layer provides different application services.

Furthermore, a number of works (e.g., [49, 235, 531, 325, 14]) have proposed a five-layer architecture that includes the following:

- Objects Layer: This is also referred to as the "Device Layer". The objects layer consists of sensor devices and physical objects.
- Object Abstraction Layer: This can be called "the network layer" or the "transmission layer". The object abstraction layer transfers information securely from the object layer (e.g., sensor devices) to the information-processing system. The transmission medium can be wired or wireless and depending on the sensor devices, the technology can be 3G, WiFi, Bluetooth, ZigBee, etc.
- Service Management Layer: This layer is responsible for the management of the services. The service management layer receives information from the network layer and stores it in a database. It then performs information processing and makes automatic decisions based on the processing results.
- Application Layer: This provides global application management based on the information about objects processed in the middleware layer. Examples of IoT applications include Smart Health, Smart Home, Smart City, Smart Farming, etc.
- Business Layer: This is responsible for managing the IoT system as a whole, including applications and services. Based on the data received from the

application layer, it builds business models, graphs, flowcharts, etc. The IoT technology's real success also depends on good business models. This layer will help to determine future actions and business strategies based on the analysis of the results.

The integration of IoT devices and Edge and Cloud layers has been considered by some studies, such as [146]. The main purpose of integrating different computing paradigms is to increase performance, enhance energy efficiency, improve the response time, and ensure better localised accuracy for future IoT and Cyber-Physical System (CPS) applications. Therefore, in our work, we attempt to consider an IoT architecture as our reference architecture, with the architecture mainly consisting of the following layers (Figure 2.1):

1. IoT Devices Layer: This layer consists of devices for sensing and reflecting the physical world, such as sensors, actuators, cameras, and smart mobile devices.

2. Edge Computing Layer: This layer pushes the intelligence (computation) to the edge of the network to improve the performance and reduce unnecessary data transference to Cloud datacentres. Moreover, the computing capacity of Edge resources is increased continuously, which allows independent decision making to take place on the Edge. Edge resources also contain sensitive personal and social data, which means that the management and control of the data flow must be moved to the Edge so that it can be managed in a more secure and private manner.

3. Cloud Computing Layer: This layer provides both hardware infrastructure and programming models (e.g., streaming and batch processing) for Big Data [2].

    - Big Data Programming Models layer: because of the Cloud's capabilities to deal with the large volume of data generated from various resources and at different rates, this layer consists of the following components, as described in [399]:

        – Data ingestion: accepts data from multiple sources such as online services or back-end system logs.

[2] Big Data has three main characteristics: Volume (large volume of data), Velocity (real time, near to real time) and Variety (different type of data: messages, sensor data, images,..)

  - Data analytic: consists of many platforms including stream/batch processing frameworks, and scalable machine learning frameworks that ease the implementation of data analytic use cases, such as Smart City applications on Cloud and Edge data centres.
  - Data storage: to store intermediate or final datasets. The ingestion and analytic layers make use of different databases during execution and where required persist/load the data into/from the storage layer.
- Cloud Infrastructure Layer: provides the consumer with processing capabilities, access to networks, storage and other basic computing resources. It enables the service user to run arbitrary software, such as applications and operating systems [495].

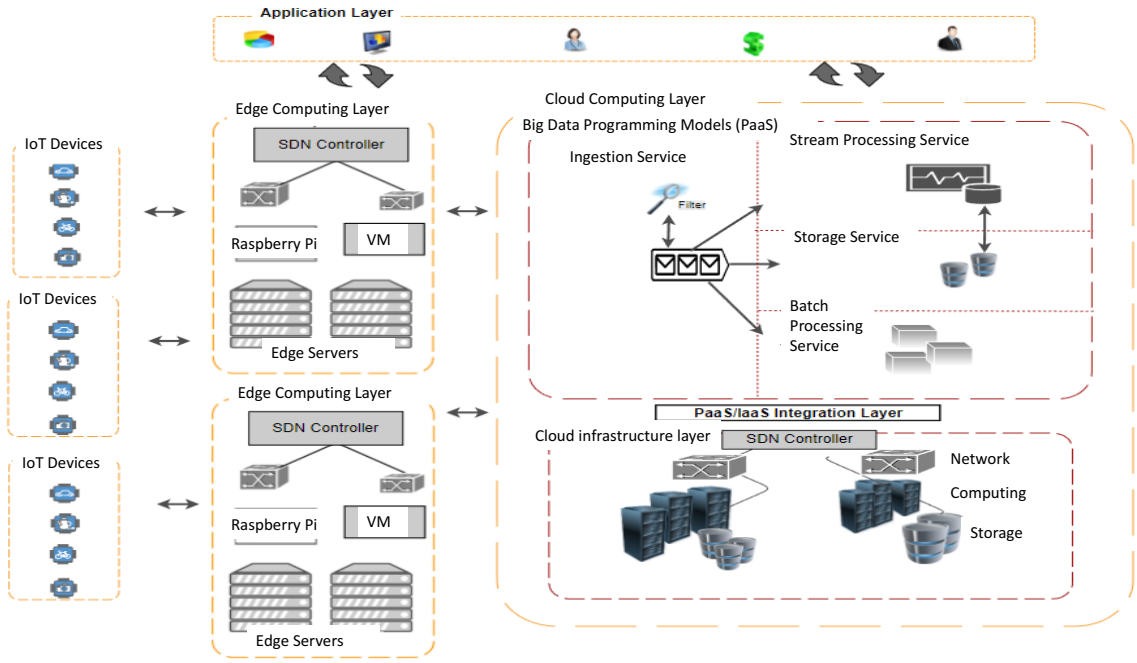

Fig. 2.1 Reference IoT Architecture

2.1.1 Service Level Agreement (SLA)

An SLA is defined by [495] as follows: "An explicit statement of expectations and obligations that exist in a business relationship between two organisations: the service provider and customer". SLAs must include a guarantee of the quality of the service, and an indication of the actions that will be required if this

guarantee is violated [432] [262]. SLAs increase the level of trust between the service consumer and the service provider. Consumers are assured that a certain level of quality is guaranteed, and if this level is not met, then they will receive compensation for any damage suffered as a result [432] [262].

Over many years, SLAs have been used in many IT-related fields and platforms [77]. An SLA passes through different stages, and these stages represent the SLA lifecycle [495]. Ron and Aliko [444] illustrate the SLA lifecycle in three stages as shown in Figure 2.2: 1) The creation phase, which allows consumers to find a provider that matches their requirements; 2) the operation phase, in which the consumers have a read-only view of the agreed-upon SLA; 3) the removal phase, during which the SLA is terminated and removed from the system.

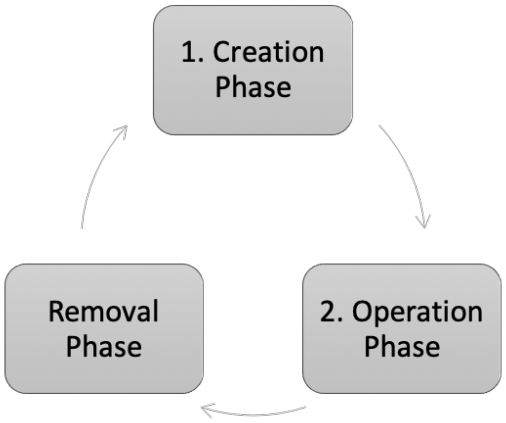

Fig. 2.2 SLA high-level lifecycle stags [432].

The Sun Microsystems Internet Data Centre Group (2002) [327] defines the SLA lifecycle in more detail by expressing it in six phases as shown in Figure 2.3. The first step is to discover a service provider by finding one that can offer services that match the consumer's requirements. The second step is the identification of facilities, groups, penalty policy, and QoS criteria for defining SLA terms. In this process, a mutual agreement can be reached between the parties. The third step is to establish an SLA, in which an agreement is formed and the parties begin to commit to the terms of the agreement. The fourth step is "monitoring SLA violation", in which the performance of the provider is assessed against the agreement terms. The fifth step is "Terminate SLA", in which the SLA ends because of a timeout or a breach of any term. In the sixth section,

"enforce-penalties for SLA violation", the relevant penalty clauses are applied and enforced if any party violates a term of the agreement.

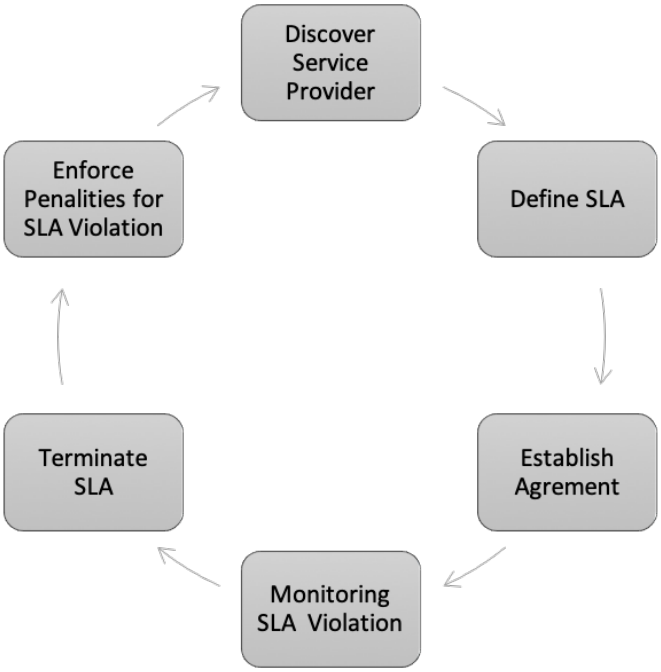

Fig. 2.3 SLA lifecycle six steps [432]

2.1.2 Blockchain

In its simplest form, a Blockchain can be defined as a distributed data structure (ledger) that transparently and securely holds transactional records. The data stored in a Blockchain network is fully open and available within the network. Moreover, once the data is added to a Blockchain, any changes are extremely difficult or almost impossible to make, so Blockchains are very secure networks.

The first commonly used Blockchain platform was the cryptocurrency Bitcoin, which was created under the pseudonym of Satoshi Nakamoto [351]. Several other cryptocurrencies have emerged since Bitcoin evolved, including Ethereum and Litecoin. Blockchain is not limited to cryptocurrencies; it can also be used in other domains such as health records, supply chains and asset ownership [206]. The information stored on a Blockchain can take different forms (e.g. money transfer, ownership, object ID, prices, etc.) depending on the technology

application. The data stored on Blockchain are open to all Blockchain network users. To add a new transaction to the Blockchain, a group of users must verify it and a consensus should be reached among them.

Essentially, Blockchain consists of chains of blocks in which each block holds information or data, its hash key and the hash key of its previous block. A hash is a fixed-length data fingerprint and it is created using a special function (called the hash function). The hash keys provide security and integrity for the data stored inside the blocks in a Blockchain network. If data within a specific block are modified, the hash of that block is also modified. This makes it easy to detect fraudulent malicious behaviours within a Blockchain network. Transactions within a block are authorised only if the hash stored in that block is correct.

If a single node within the Blockchain network wants to transfer a transaction, the sender generates a block containing information such as a digital signature, timestamp, and the public key of the recipient. The information block is then broadcasted through the network. After that, the validation process begins, which is one of Blockchain's most significant features. Validation is the phase where transactions are validated to prevent malicious data alteration [521]. Validated transactions are listed to be appended to the Blockchain using a consensus protocol. Every node validates transaction and user status. When checked by the majority of nodes in the network, the block is timestamped and added to the current blockchain. Eventually, Blockchain's current copies are updated to reflect network changes.

Consensus protocol is one of Blockchain's most important aspects, as it helps to create an irrefutable system of real-time agreement between various users of a universally shared ledger. Owing to the Blockchain's decentralised nature, no centralised authority verifies updates to the ledger or new transactions. Users have to agree among themselves which transactions are to be added to the Blockchain. But how can thousands of users distributed around the globe reach such an agreement? This is where a consensus algorithm comes in. To achieve a consistent shared ledger state, all of the participants in a decentralised network must follow the protocol. Different Blockchain implementations use different consensus protocols to achieve a shared ledger. Proof of Work (PoW), Proof

of Stake (PoS) and Byzantine fault tolerance are the most popular consensus protocols used by the major Blockchain applications such Bitcoin, Etherium, and Monero.

2.1.3 Smart Contract

A Smart Contract is a decentralised transactional protocol enforcing the terms of a contract with the intention of satisfying common contractual requirements between the parties involved [393].

The idea of smart contracts was initially coined by Szabo in 1994[3]. Szabo claimed that the hardware and software could be linked to a number of contractual clauses in a way that would make the violation of a contract very costly. Although the concept of smart contracts had existed for decades, smart contracts gained the publicity we see today only after the advent of Blockchain technology. Blockchain allows a set of rules to be implemented on a distributed ledger in the form of a computer program and it implements and enforces the terms of agreements automatically. Blockchain-based smart contracts make it possible to exchange items in the form of money, shares, properties, etc., quickly, transparently, and cheaply between different parties. In addition, Blockchain-based smart contracts eliminate the need for trusted intermediaries like banks, attorneys, advisors etc. In addition to implementing the contract terms and conditions specified in the agreement, smart contracts are capable of carrying out other activities, such as collecting data from outside the Blockchain and processing it according to the contract terms [259].
In general, smart contracts, also called Cryptographically Enabled Contracts (CryptoECs) [149], work for digital asset transactions with multiple participants who can automatically handle properties. Assets can be distributed among participants according to the rules stated in the contracts. Smart contracts are performed in real time and are self-enforcing and tamper-proof due to their decentralised nature.

[3] http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html

2.2 State of the art

In this section, the focus is on providing an overview of available SLA studies, which are classified from a technical point of view. However, with the limited work on SLAs for IoT, previous works, especially works related to Cloud computing, have also been considered. The reason behind considering Cloud-related works is the dependency between the IoT and the Cloud. We have applied a systematic review approach to surveying the published works. The following section presents this approach and the survey results.

2.2.1 Research Methodology

We followed the systematic mapping study discussed in [384] due to its clarity and easy-to-follow presentation. The purpose of applying a systematic mapping study is to explore available research related to SLAs for IoT applications. The steps for applying the systematic mapping study are presented in Figure 2.4. Each step has an outcome and the outcomes from one step are inputted into the next step. The final outcome of the systematic mapping process enables us to identify the research area and the research gaps related to SLAs for the IoT.

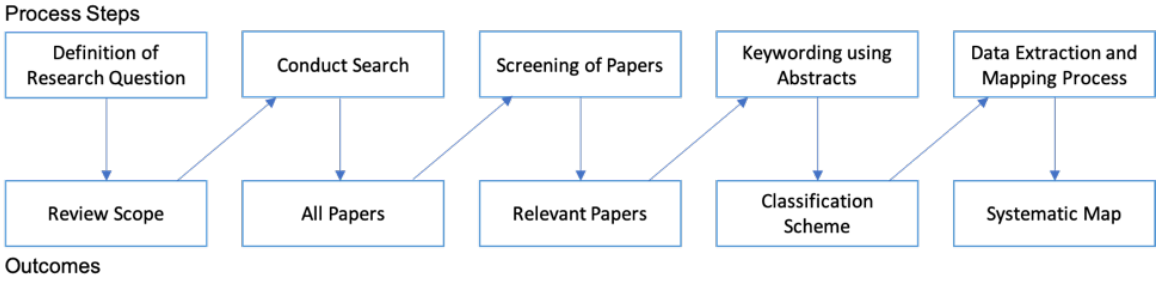

Fig. 2.4 The steps of the systematic mapping process [384]

In the following we provide details about each step of the systematic mapping study (depicted in Figure 2.4) and how it can be applied:

- Step 1: Definition of Research Questions - The main purpose of this step is to define the research questions in order to guide our research process towards answering them:
  - RQ1. What are the current research topics related to SLAs for the IoT?

  - RQ2. How active are research topics related to SLAs? (Measured by capturing the number of publications in the last 10 years)
  - RQ3. What is the number of publications per year related to SLA specifications?[4]
  - RQ4. What is the number of publications per year related to SLA-aware service placement?[5]
  - RQ3. What are the research gaps that need to be addressed in future studies?

- Step 2: Conduct Search - In this step, we searched scientific databases for all relevant scientific research papers using search strings. We selected IEEE, Science Direct, ACM Digital Library and Springer as scientific databases due to their high scientific impact. The keywords that we used to collect the relevant resources are presented in Table 2.1:

Table 2.1 Keywords of the Search

Row No.	**Keywords that are used for the search**
I	SLA? OR Service Level Agreement?
II	Cloud Computing OR Cloud OR Internet of Things OR IoT OR Edge OR Fog

  - First, to retrieve a broader result and determine the classification criteria that should be used to classify the available research, we built the query string, which is comprised of the terms in the first row of Table 2.1.
  - Second, to narrow our research results we utilised the first row and the second row, joining the keywords with the AND connector.

  We used the ? wildcard to retrieve relevant topics regardless of whether they were in the singular or plural form.

- Step 3: Screening of Papers - In this step, we aimed to select the papers that were related to our topic of interest, so we followed the following steps: the title of the paper was considered to be the first indicator of whether

[4]This is because we are interested in the SLA specification phase
[5]This is because we are interested in SLA-aware service placement for further research

or not the paper should be considered. If the title did not indicate an approach related to our topic, then we read through the abstract, checked its contribution and discarded any studies that did not contribute to our research questions. If the abstract was not enough to indicate the relevance of the paper, we examined the paper by reading its content.

- Step 4: Keywording using Abstracts - In this step, we read the abstract to identify the most important keywords that reflected the contribution of the research. If the abstract was not enough to indicate the relative keywords, then we read through the paper. After identifying the keywords, we clustered them into categories in order to form the classification scheme for mapping purposes.

- Step 5: Data Extraction and Mapping Process - This step allows the reviewers to answer the research questions. Thus, we gathered different data items from each selected scientific publication to understand its main aim and contributions to answer the research questions.

2.3 Results

In this section, we discuss the criteria that we apply for screening the search results, as well as the classification mechanisms.

We applied our search strings as discussed in Section 2.2.1 for different scientific databases. After that, we applied the screening step to select relevant papers. First, we managed to collect 2,511 papers. Then, we excluded papers that displayed one or more of the following characteristics: repeated (i.e., appeared more than once in the search results); not written in English; too general (i.e., presented knowledge about SLAs in general); or the focus was not on SLAs. As a result, the number of considered papers was reduced to 288 and to 124 for work related to the SLA lifecycle class and SLA applications class, respectively. Figure 2.5 shows the searching and screening results. We found that a considerable number of works could fall into the following categories (see Figure 2.6).

- SLA Lifecycle Category: This consists of works whose main contribution is focused on one phase or more of the SLA lifecycle. This category consists of the following subcategories: 1) SLA specification; 2) SLA negotiation; 3) SLA monitoring; 4) SLA enforcement; 5) SLA management; 6) Others.

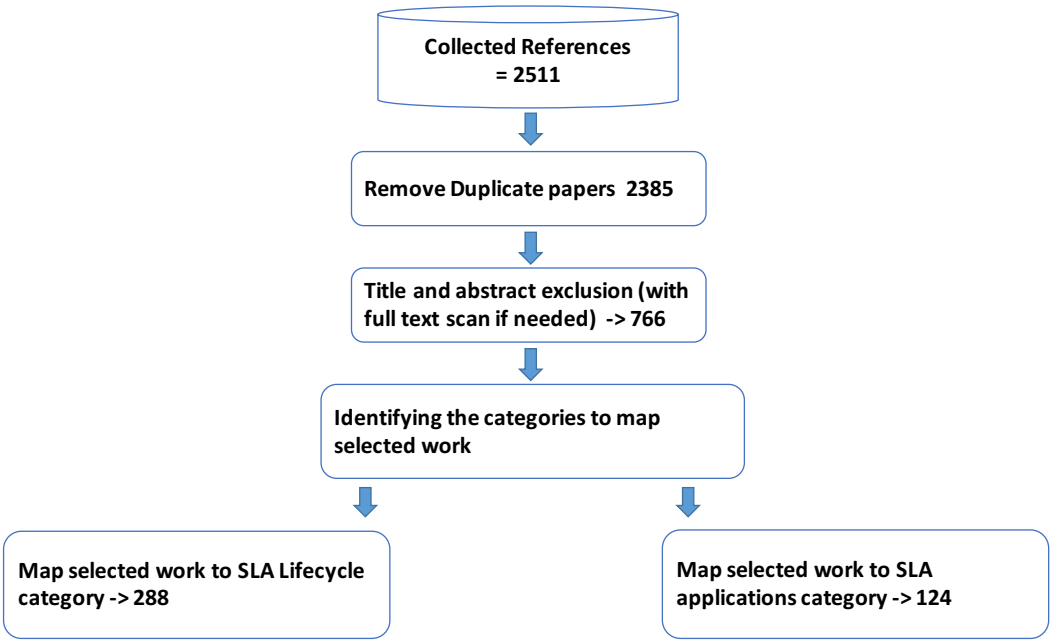

Fig. 2.5 Study selection step of the systematic mapping process

- SLA Applications Category: This includes works that provide solutions, mechanisms, and policies to deliver the required functionalities while considering the SLA constraints. This category consists of the following subcategories: 1) Scheduling; 2) Load balancing; 3) Elasticity; 4) Resource provisioning and allocation; 5) Resource management; 6) Service placement; and 7) Orchestration.

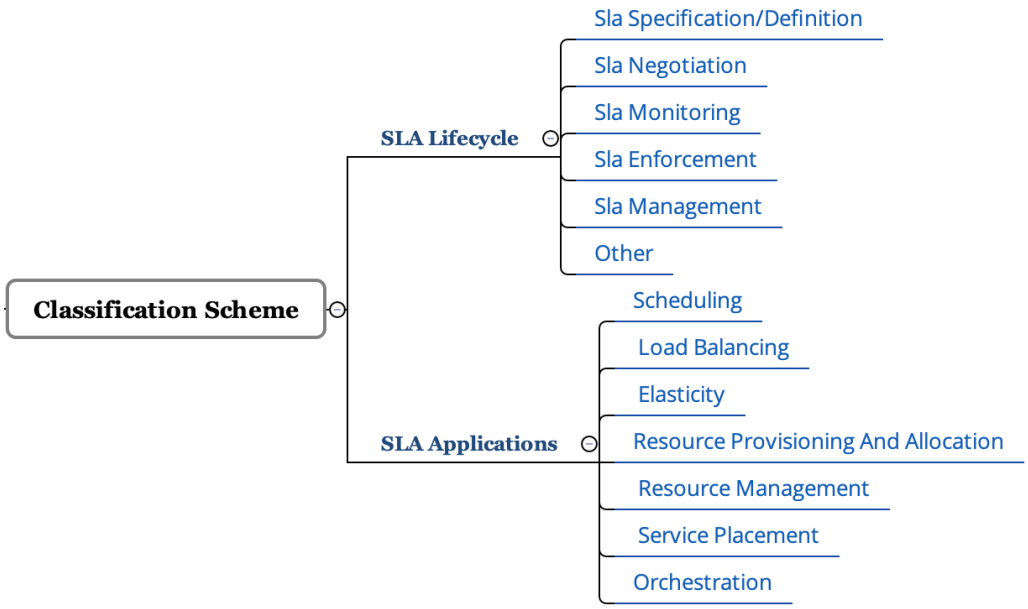

Fig. 2.6 Main categories based on our conducted search in relation to SLA

The results show that about two-thirds of the papers focus on the SLA lifecycle while the remaining papers focus on SLA applications.

In the following sections we present an overview of some works that are mapped to the subcategories of the two categories reflected in our classification scheme presented in Figure 2.6:

2.3.1 Works Related to SLA Lifecycle Category

From the literature, we found many studies whose main contribution focuses on one phase or more of the SLA lifecycle. Table 2.2 maps the selected work to the following sub-categories: SLA specification, negotiation, monitoring, enforcement, and management.

Table 2.2 Results of mapping published studies to SLA lifecycle phases

SLA Specification	SLA negotiation	SLA Monitoring	SLA Enforcement	SLA Management	Others
[13] [256] [245] [169] [211] [217] [443] [172] [189] [24] [233] [36] [255] [421] [475] [333] [422] [460] [196] [246] [268] [153] [282] [283] [448] [446] [15] [238] [252] [228] [38] [227] [476] [190] [412]	[163] [301] [300] [86] [427] [194] [129] [349] [387] [362] [90] [204] [298] [16] [295] [57] [510] [373] [438] [436] [253] [363] [312] [191] [31] [530] [324] [78] [5] [13] [498] [120] [142] [398] [145] [439] [34] [275] [369] [266] [174] [368] [130] [58] [270] [371] [182] [403] [17] [339] [440] [388] [389] [197] [328] [91] [297] [502] [358] [249] [114] [181] [175] [451] [450] [72] [330] [296] [119][267] [198] [207] [120]	[469] [334] [434] [78] [199] [205] [209] [232] [285] [318] [336] [404] [406] [417] [467] [215] [341] [200] [380] [124] [20] [355] [62] [122] [111] [185] [419] [164] [56] [350] [326] [411] [461] [18] [254] [470] [294] [291] [281] [383] [211] [173] [113] [217] [74] [201] [405] [102] [155] [382] [123] [397] [23] [424] [172] [25] [50] [335] [367] [152] [180] [465] [7] [144] [414] [219] [234] [523] [52] [258] [103] [457] [4] [534] [485] [134] [522] [37] [51] [462] [73] [490] [126] [410] [75] [486] [193] [26] [449] [41] [354] [356] [161] [162] [365] [412] [151] [252] [116] [184] [228] [38]	[476] [9] [299] [390] [300] [416] [469] [302] [334] [477] [395] [10] [532] [434] [220] [36] [257][319] [301]	[374] [220] [287] [407] [209] [232] [336] [526] [88] [411] [23] [19] [255] [421] [333] [153] [278] [319] [221] [525] [473] [257] [72] [464] [156] [519] [330] [449] [41] [43] [71] [497] [524] [138] [247] [420] [81] [84] [446] [157] [322] [354] [356] [361] [319] [228] [192] [340]	[250] [454] [222] [456] [455] [516] [394] [251] [331] [332] [432][447] [342] [136] [435] [125] [344] [292] [158] [208]

SLA specification/definition:

This sub-category includes works related to the SLA specification, mostly providing/defining the SLA template/language to express the consumers' requirements and the providers' capabilities. The SLA definition is an important step in mitigating the risk of ambiguity and high expectations. Based on our survey, we mapped 35 published works under the SLA specification category. This section provides a more detailed description of the available works related to SLA specifications:

Specifying and monitoring SLAs in the grid have been studied by [412] where they propose an architecture for specifying and monitoring SLAs. To formalise SLAs, they specify an SLA based on measurable data constrained by date (start date, end date) and a set of service-level objectives. SLA@SOI [252] is a framework to address multi-level multi-provider SLA lifecycle management within service-oriented architecture and Cloud computing. Within the SLA specification stage, SLA@SOI provides a service description template for SLA which is called Service Level Agreement Template (SLA(T)). SLA(T) model is a language and technology independent model and it can be modelled using XML, OWL, human readable language or any concrete syntactic format. It provides an abstract syntax for describing the functional and non-functional characteristics of a service. However, there is a need to add more domain-specific vocabularies related to the domain it describes [472]. Thus, we aim, in our work, to consider more domain-specific vocabularies for the IoT to allow for specifying requirements at fine-grain details level.

The Web Service Level Agreement (WSLA) framework [228] comprises an SLA specification and a number of SLA monitoring services. The WSLA framework provides a WSLA specification that is specified using XML language and it consists of three sections: 1) Parties: the service consumers and the service provider are called the signatory parties, while other external agents such as third parties are called supporting parties. The signatory party descriptions include identification and technical properties such as address. Supporting parties have additional attribute to indicate the sponsor of the supporting party. 2) Service Description: to specify the features of the service and its parameters. 3) Obligations: to define the guarantees and constraints of the SLA parameters. WSLAs have several key features: the metrics are flexible (supporting composite metrics for example) and

extensible, and WSLAs contain extensive documentation. Importantly, WSLAs have been structured so that the contractual and monitoring clauses are separated; this means that a third party can be employed to provide the service without giving them access to sensitive information [472].

On the other hand, WSLAs have some weaknesses [472]. There are formal semantics for the language [495]; it is linked to the monitoring infrastructure in the commercial solution [495]; it lacks reusability and is based on the XML-Schema, the semantics of which are not appropriate for constraint-oriented reasoning and optimisation [227] [472]. Furthermore, despite the online monitoring and contracting techniques employed by WSLAs, under which circumstances the service objective is violated is not clearly specified [22].

The web service agreement (WS-Agreement) from the Open Grid Forum (OGF) defines a web service agreement specification as a protocol for launching an agreement between two parties, including an agreement template to aid the discovery process for well-suited agreement parties. The objective of the WS-Agreement specification is to define a language and a protocol for advertising the capabilities of service providers, creating agreements based on creational offers and monitoring agreement compliance at run-time [38]. Specifying the WS-Agreement serves several purposes: it defines a language and protocol so that service providers can advertise their capabilities; it creates agreements based on creational offers; and it monitors compliance with the agreement at run-time [38]. However, similar to WSLA, WS-Agreement is also linked to XML-Schema, the semantics of which are not suitable for constraint-oriented reasoning or the optimisation demands of operation research [227]. Furthermore, WS-Agreements relate to web services, while in an end-to-end SLA specification of the IoT approach, the specifications of the Cloud, network and Edge layers are required. Nevertheless, WS-Agreement only provides an SLA specification for web services at a high level (e.g., details about the contractual parties and the start and end dates of the agreement). It leaves the fine-grained content unspecified [472]. This might cause ambiguity due to the possibility of having different definitions among the involved parties .

SLA* is an SLA language that provides finely detailed specifications of SLA content, making it extensible and extremely expressive. It is seen by many as another promising language [227]. It was developed to be a generalised and refined form of the web-service specific XML standards: WS-Agreement, WSLA, and WSDL. Rather than dealing with just web services, SLA* deals with services in general, and it is not dependent on any language. It can be used to support a wide range of functions from controlled customisation to arbitrary domain-specific requirements. Furthermore, the SLA* framework can be applied to any phase of the SLA lifecycle and it has been tested in a variety of domains. The disadvantages of SLA* are that its multi-domain approach means that it does not possess precise semantics and it cannot support brokerage. Furthermore, a specific vocabulary needs to be developed for each domain [472].

It is increasingly common for applications or web services to be utilised across organisational boundaries, with new services appearing at the network and storage levels. Industry is also making an increased use of languages to specify interfaces for such services. End-to-end QoS has been researched and it has been confirmed that the provision of QoS is multi-faceted, requiring complex agreements between network services, storage services, and middleware services [256]. SLAng [256] is a language for defining SLAs which is introduced to meet these needs. SLAng includes: 1) an end-point description of the contractors, such as the location or facilities of the provider or customer; 2) contractual statements such as when the agreement starts, how long it will last and the charges that will apply; and 3) Service Level Specifications (SLSs), i.e. the technical QoS description and the associated metrics.

The strengths of SLAng lie in the fact that it utilises a domain-specific vocabulary related to IT services and Application-Service Provision (ASP). It also emphasises SLA compatibility, monitorability and constrained service behaviour [472]. The weaknesses of SLAng include the fact that the domain-specific QoS constraints are limited, it focuses on electronic services [227], and it has not been updated since 2009 [432]. Furthermore, it is very complex [472]. It is difficult to gain a full understanding of its specification and then to use this specification to generate SLAs and extend the language. Technical experts are needed to employ it due to its formal nature and the way it combines techniques

such as Object Constraint Language (OCL) and Essential Meta-Object Facility (EMOF). Real-world cases require extensions that are similar in size to the SLAng language itself [432]. All of these factors mean that SLAng is not only difficult for users to employ, but it is also expensive.

Cloud Service Level Agreement (CSLA) [245] is a specification language developed specifically for the Cloud domain presented in [245]. There are new features in CSLA, for example QoS/ functionality degradation and an advanced penalty model, which means that providers are able to determine the fine details of contracts in order to improve the self-adaptation capabilities of services and minimise SLA violations. In terms of structure, CSLA has many similarities to WS-Agreement. The dates that the SLA begins and expires are described under validity. The parties section of the agreement defines the parties involved, while the services, constraints, charges and termination conditions are defined in the template [472].

CSLA provides novelty because it facilitates pay-as-you-go charging models, as well as conventional fixed-price charging [472]. It also introduces fuzziness and Confidence concepts. fuzziness determines the margin of error for the metrics included in the agreement, and confidence. Confidence provides a definition of the minimum ratio by which the metric values are allowed to surpass the threshold, although the fuzziness threshold cannot be exceeded. For example, if the threshold for a service's response time is set to 3 seconds, the fuzziness value is 0.5 and the confidence is 90%. For every 100 requests, at least 90 requests should be between 0 and 3 seconds and no more than 10 requests can be between 3 and 3.5 seconds, otherwise the SLA is violated. However, CSLA is not formally defined and some parties who have significant roles, such as the broker, are not supported [472].

Another example is Service-Level-Agreement Language for Cloud Computing (SLAC), which is defined by [476]. The language is developed for specifying SLAs in the Cloud computing domain. It is differentiated from previous specification languages in that it is domain specific, facilitates multi-party agreements, supports the principle of Cloud deployment models and has formally defined semantics. The business elements of Cloud computing are also supported, such

as pricing schemes, business actions, and metrics. Even SLAC provides domain-specific vocabularies for Cloud computing. However, the focus of SLAC is only related to Infrastructure as a Service (IaaS) [476], whereas we aim to consider SLA specification on end-to-end basis. End-to-end SLA allows us to consider all of the parties/components involved in an IoT ecosystem.

Al Falasi and Serhani [13] address SLA specification and negotiation difficulties in a federated Cloud network, CloudLend. They suggest a weighted SLA specification model to capture the QoS of clients and to manage the specification of multi-level SLAs. For each Cloud service in CloudLend, there is a public SLA profile that includes the following specifications: information related to the Cloud service, and information related to QoS terms and their specified weight. The weight specifies the percentage of the values that a CloudLend member preserves for each SLA term and its specified parameters. Furthermore, for each established relationship between two Cloud providers, the SLA captures the following details: information related to both services, including name and type of service, the provider and a reference to the service; information related to the agreed-upon relationship, indicating reference, type, initiator and attendant service, the start time of the relationship, the duration of validity, and QoS terms. However, CloudLend is developed for the Cloud computing paradigm and there is no formalism of the specification. Furthermore, the specification is in an XML format, whereas there are other formats such as JavaScript Object Notation (JSON) which can be more human-readable as well as more lightweight.

Although the above-mentioned works are very close to the topic that we are aiming to investigate, i.e. SLA specifications, a considerable number of works are relevant in some way to SLA specification and modelling. For example, Cloud Security-SLA standardisation [190] enables a simple and automated assessment of the safety products of service suppliers. It also assists Cloud clients to perform a more guided decision-making and selection process. The presented work in [190] suggests a new strategy for building the normal Security-SLA format based on the security services usually supplied or to be supplied in Cloud environments. It focuses mainly on the protection of the essential security requirements, confidentiality, integrity and availability, as the primary goals of the security

agreement.

For the formalisation and management of SLA information, Stamou [446] suggests a directed property-graph data model. The presented work follows the semantics and structure of the WSLA specification presented in [228], which means that it is an XML-based format to serve the purpose of information portability. Klingert [238] presents the GreenSLAs concept, which is described as an SLA between a datacentre and its consumers, to provide more eco-efficient operations than those provided under traditional performance-based SLAs. If the data centre provided all the facilities in the most eco-efficient way (technically feasible), neglecting the effect on QoS, GreenSLAs would not be required. Thus, a GreenSLA is provided along with regular SLAs, using co-efficiency as a key differentiation factor. However, it is an XML-based SLA since it follows the WSLA format where we are looking for more compact serialisation for transmission or mass storage.

Another study is presented in [283], which extends the XML-based WSLA framework in [228] in order to provide a flexible template for IT service contracts. The proposed work aims to overcome two main issues in SLA management: the lack of standard models to represent service contracts and their associated SLAs in service-oriented architecture; and networking environments. These are resolved by extending the WSLA framework. Another issue is providing a machine-readable SLA and this is overcome by modelling the template as a digraph, which is implemented using a NoSQL graph DBMS.

The study presented in [196] introduces a schema to manage the green energy of data centres. One of the main contributions presented in [196] is extending CSLA [245] to support a Green SLA by adding two threshold parameters (minimum and maximum thresholds). However, it inherits CSLA's weaknesses as there is no formalism and it is an XML-based SLA . Li et al. [268] propose a PaaS Level SLA Description Language (PSLA), a well-structured description language based on WS-Agreement. The PaaS Level SLA summarises and defines the semantic clauses that need to be considered, in particular specific feature of PaaS such as work elasticity. However, [268] considers the PaaS tier only, whereas we are interested in considering an SLA on an end-to-end basis.

In the Edge datacentre, there is lack of research on SLA management. However, [307] reviews some works within the literature under the Fog computing paradigm, where data in Fog computing are handled at the edge of the network. The reviewed works consider different service-level objectives (SLOs), which have been achieved by moving intelligence to the edge of the network. Examples of SLOs are: network management as in [117, 140], resource management for the IoT as in [1, 183], and latency management in 5G cellular networks as in [370, 212]. Furthermore, some research extends the WSLA specification to consider devices (sensor devices) as a fourth section that can be listed in the SLA specification, as in [165]. However, this work did not specify any related aspects of Cloud computing services, such as data storage or analysis, since its focus is on wireless sensor networks.

SLA negotiation:

This section includes works concerned with negotiating SLA terms between services and consumers in order to reach an agreement that satisfies the involved parties. There is a considerable amount of research that investigates SLA negotiation; we have mapped 74 works to the SLA negotiation sub-category. For example, Baig et al. [57] propose a formal model of SLA-based negotiation to enable a multi-round SLA negotiation that can adapt to a variety of client requirements, pricing models, and decision strategies. This work is developed using the WASAG4J library, which is part of WS-Agreement implementation [38]. Nevertheless, [57] is applied to support real-time bilateral negotiations for Cloud services.

Yaqub et al. [510] tackle the inflexibility problem of the available take-it-or-leave-it SLAs by proposing a robust and inexpensive negotiation method that can create near-to-optimal SLAs, considering the time constraints. The proposed work allows for providing a dynamic SLA negotiation that was evaluated; the experiments show an improved level of participant satisfaction. Al Falasi and Serhani [13] address SLA specification and negotiation difficulties in a federated Cloud network, CloudLend. They implement an independent model of SLA negotiation that adopts an improved game of fair division. A number of studies have considered multi-issue negotiation [436, 363, 439, 297]. For example, [436] and

[439] propose an interactive SLA negotiation strategy to support multi-criteria negotiation, including negotiations about time slots and prices. Furthermore, other studies that are interested in SLA negotiation propose a multi-stage SLA negotiation mechanism, such as [198, 369, 427]. However, all of these studies apply to Cloud computing, which means that their practicality have not been tested for IoT applications, where there is a greater possibility of having multi-party agreements.

Additionally, a number of studies have addressed negotiation and renegotiation approaches, such as [194, 72, 197]. For example, Hashmi et al. [197] present a framework for web service negotiation that would be used by both consumers and web service providers to automate negotiations for the quality level of web services. It is a flexible framework that is protocol independent and it supports communication, negotiation, and SLA development based on participant policies. In a multi-service and multi-party negotiation scenario, it allows multi-round negotiation for multi-criteria negotiated service modelling since it extends WS-Negotiation [207] and WS-Renegotiation [197]. However, the proposed protocol is applied to the web service paradigm.

Applying optimisation solutions to SLA negotiations has been studied by a number of researchers. For example, Abulkhair et al. [5] apply a parallel implementation of particle swarm optimisation in order to enhance SLA negotiation in the Cloud, aiming to reduce negotiation time while increasing throughput. Furthermore, Copil et al. [120] design an SLA negotiation mechanism to provide a balance between the consumed energy and the offered performance in the Cloud by applying particle swarm optimisation techniques. Maity and Chaudhuri [312] apply multi-objective genetic algorithms to provide an optimal negotiation of the SLA in a federated Cloud environment.

The current SLA negotiation literature is limited, especially when considering large-scale, dynamic environments such as the IoT. The negotiation protocol is a crucial area of the SLA negotiation phase and is discussed in various Cloud projects. However, existing IoT negotiation strategies follow a centralised approach, which may not be realistic, given the dynamicity and distributed nature of the IoT environment [267]. However, Li et al. [267] suggest a negotiation

process using decentralised network brokers to negotiate efficiently on behalf of service customers with multiple IoT service providers. The framework uses a hierarchical architecture to manage the message flows during the negotiation process and to cluster service information. Nevertheless, there is a need for further research to consider an efficient, dynamic negotiation protocol. Furthermore, Zheng [530] implements a mixed approach to Cloud service negotiation in the IoT environment based on the "chicken game". However, the focus is related to negotiating Cloud services.

SLA Monitoring:

This sub-category includes works whose main focus is SLA compliance (by detecting a violation or predicting a violation before it occurs). We have mapped 103 out of 288 works to the SLA monitoring sub-category. Table 2.2 shows the mapped references; most of the publications are related to Cloud computing. For example, in [412], after specifying the SLA, the monitoring stage takes place within the proposed SLA management framework [412]. This allows measurable metrics to be collected between the contractual parties of grid services, to evaluate the management properties of grid services in the lifecycle [21]. The authors, in the case of non-local measurement, design a measurement-exchange protocol to minimise the cost of the transmitted data, taking into account the time at which the SLA evaluation is triggered [412]. Although this work monitors SLAs by collecting measurements between different contractual parties, which is similar to the idea of having multi-providers in an IoT application. However, the specification of their work still focuses on grid requirements.

Cloud providers tend to use available monitoring tools with some adaptations. These tools, in most cases, are designed for a homogeneous environment, are not scalable, and do not provide a mapping facility from low-level resource metrics (e.g. mapping from uptime/downtime) to high-level SLA parameters (e.g., the availability objective) [151]. Emeakaroha et al. [151] aim to monitor and enforce SLA objectives in the Cloud environment, in particular scalability, efficiency and reliability requirements. They provide an LoM2HiS Framework, which aims to map low-level resource metrics to high SLA objectives. LoM2HiS is part of the Foundation of Self-governing ICT Infrastructure (FoSII) research project at Vienna University of Technology. Each FoSII service has three interfaces: the

negotiation interface, the job-management interface and the self-management interface to prevent SLA violations. FoSII has two components: the Enactor Component and the LoM2HiS Framework.

In the LoM2HiS Framework, when the SLA objectives have been agreed upon, the mapped rule from the agreed SLA, using domain-specific language, will be stored in an agreed SLA repository. When the customer makes a resource-provisioning request, the run-time monitoring will load the service SLA from the agreed SLA repository, and then monitoring agents will monitor the resource metrics, which can be accessed by host monitoring. The host monitoring will transmit the extracted metric-value pairs, periodically, to the run monitoring and to the enactor component using the designed communication mode. The main contribution of this work is having the mapping stage between the low-level metrics and the high-level SLA. However, it is applied only to monitor the infrastructure layer.

SLA@SOI also provides a three-layered SLA-driven monitoring framework [252]. There is a sensing and adjustment layer (lower layer) to collect the events using a reasoner (monitor), which has the ability to understand SLAs and then implement monitoring rules using abstract syntax trees. The research in SLA@SOI is driven by four use cases: ERP Hosting, Enterprise IT, Service Aggregation, and e-Government, but none of these are IoT-based use cases.

Cicotti et al. [116] provide a QoS monitoring for Cloud IaaS (QoSMONaaS). QoSMONaaS is designed and implemented in such a way to allow for event collection, event-pattern recognition, and event correlation using Complex Event Processing to satisfy high-performance requirements, as well as working in a dynamic and heterogeneous environment. QoSMONaaS monitors QoS at a business level, not only network and/or Cloud resources. The authors present the concept of Quality Constraint (QC), which for each single Key Performance Indicator (KPI) uses a Boolean condition. Here the KPIs are specified within the SLA, along with the time interval within which the KPI will be measured.

The QoSMONaaS framework [116] consists of two layers: a business logic layer to perform QoS checking and monitoring and a data layer, which includes

a Complex Data Processor (CDP). The CDP is used for real-time data stream analysis and processing, as well as for storing data related to SLAs in a DBMS for managing SLA negotiation, registration, and monitoring. This work provides monitoring as a service, which gives Cloud users the ability to use it seamlessly. It also takes into consideration the high-performance requirement that is essential in real-time systems. This means that its implementation could be of benefit for QoS monitoring applications in the IoT paradigm.

Furthermore, SLA monitoring is essential to ensure that the service is delivered according to specified quality levels. Therefore, many studies have been conducted on QoS/SLA monitoring, such as [228, 38, 412]. They provide SLA monitoring in web services and at the Grid and Cloud level, while [184] implement a distributed monitoring system for network resources. Furthermore, Cicotti et al. [116] provide monitoring as a service and support the monitoring system by using mapping rules from low-level metrics to high-level SLA requirements. However, it is applied to monitor collected data of Cloud infrastructure tier.

Most of the above-presented works are related to Cloud computing. However, from the IoT perspective, there is a need to build a cross-layer multi-provider SLA-based monitoring system for the IoT. This can play a role in enhancing the end-to-end SLA adherence process of IoT applications.

SLA enforcement:

This includes works that provide mechanisms/polices to enforce an SLA. In the literature on this field, there are a number of studies that focus on investigating SLA enforcement strategies. We have mapped 19 works to the SLA enforcement sub-category, such as [220, 36, 257, 434, 532, 395, 10, 477].

A number of works aim to automate SLA enforcement. For example, Kapassa et al. [220] present a black box approach to mapping the high-level requirements to the low-level parameters defined in the infrastructure management policies, to guarantee QoS enforcement. Vakilinia et al. [477] propose a strategy to automate SLA enforcement for Cloud services by detecting and predicting events that cause SLA violations. It, first, trains a Dynamic Bayesian Network (DBN), using the collected data to calculate the dependency between different entities

across Cloud layers. Then, it feeds the correlation values into the long short-term memory neural network for prediction purposes. However, this work is applied to Cloud services where the dependencies across IoT entities are more complicated.

Madheswari et al. [302] state that SLA enforcement can be achieved using their proposed system. They propose a performance-optimised routing mechanism to be applied by a Cloud broker. Based on the mechanism, the broker can decide which data centre offers the best service for the consumer's requests. As a result, using the routing mechanism and considering different types of clients, the QoS constraints of high-priority clients can be achieved. However, the presented work is applied to the automation of SLA enforcement of Cloud services.

In addition, due to the dynamic nature of IoT and Cloud-based interactions, automated monitoring and enforcing of the service contract policies are essential. The authors Solaiman et al. [434] propose a novel model that uses business rules to represent contract terms in order to check contract compliance and enforce the contract clauses. They define which events the underlying messaging middleware needs to generate and capture, and determine important technical issues related to designing a state-aware contract monitoring and enforcement service. However, the proposed work does not consider the multi-layer nature and complexity of the IoT since the implementation does not reflect multi-level interactions. It seems that they applied it to a single consumer and a provider.

Due to the lack of a trustworthy platform, enforcing SLA in the Cloud is challenging. Zhou et al. [532] introduce a witness model to enforce the Cloud SLA terms. By implementing the witness role and using a smart contract based on Blockchain, it resolves the trust problems about who can detect the service breach, how the breach is verified and how compensation is guaranteed. In this model, in order to select autonomous witnesses to form a witness committee, a verifiable consensus-sorting algorithm is suggested. The witness committee receives payment for monitoring and detecting breaches in the SLA, thus it is essential to develop the pay-off function of the witness in the contract. Furthermore, the study utilises game theory to evaluate and demonstrate that the witness is trustworthy, in order to avoid the greedy nature that might affect the reporting

of a violation. When the witness commission confirms the service breach, the compensation is automatically transmitted to the client using the smart contract. To show the feasibility of the study, they implement a proof-of-concept prototype with the Ethereum Blockchain's smart contract. However, it is applied to enforce the trustworthiness of Cloud SLAs.

SLA management:

This includes works whose main concern is SLA management. Typically, SLA management includes more than one phase of the SLA lifecycle such as SLA negotiation, monitoring, and enforcement. We have mapped 48 works to the SLA management sub-category. For example, Keller and Ludwig [228] provide a WSLA framework that comprises the SLA specification and a number of SLA monitoring services. The WSLA framework implements 5 stages for the web service SLA management lifecycle: SLA negotiation and establishment; SLA deployment; service-level measurement and reporting; corrective management actions and SLA termination. However, although WSLA offers an adequate level of online monitoring and contracting techniques, there is no clear specification of the level at which the service objective can be described as being in a violation state [22].

In [464], the authors propose an SLA management framework for Cloud computing and inter-Cloud environments in particular. This framework is based on the WSLA implemented by IBM, but it has been altered to fit Cloud computing. Zhao et al. [525] present a new approach to the SLA-based management of Cloud-hosted databases. They present an end-to-end framework for managing Cloud-hosted databases with a consumer-centred SLA. The framework promotes the adaptive and dynamic provision of the software applications database level, based on application-defined constraints to meet their own SLA performance demands. It allows avoiding the cost of any breach of SLA and controlling the financial cost of the assigned computing resources. The framework monitors the application-defined SLA continuously and, when required, automatically triggers the execution of the necessary corrective actions (database tier scaling out/in). The framework is a platform-agnostic database which utilises processes for the replication of a database based on virtualisation. It needs a zero change of the

source code for Cloud-hosted software applications. The experimental findings show the efficacy of the proposed SLA-based system in offering the flexibility needed to meet SLA demands. However, their work only considers SLA management for the database tier. Mavrogeorgi et al. [319] present a Cloud-based SLA management system. To proactively detect and manage possible SLA violations, they propose an SLA enforcement mechanism for Cloud services based on rules, and these rules are updated in run-time.

Liu et al. [278] introduce a low-intrusion lightweight Cloud environment. The suggested Cloud platform is helpful for enabling the concept validation of new research insights and/or for conducting empirical analysis experiments based on their scalable features. The suggested SLA breach detection method is efficient and it offers important prototyping and experimentation parameters. However, it is applied for Cloud paradigm.

Moreover, as the implementation of SLAs can mitigate the potential hazards associated with availability, performance, and security in Cloud computing, Bendriss et al. [72] introduce the design of Cloud-oriented SLA services that depend on the use of a REST-based API. These services can readily be built into current Cloud applications, platforms, and infrastructures to promote the delivery of SLA-based Cloud services.

Most of the above-mentioned studies cover SLA management for Cloud Computing. Nonetheless, there is a lack of feasible SLA management systems that can standardise SLAs and which have the ability to manage all aspects, from specification, negotiation, and monitoring to the compliance of the SLA with the IoT, autonomously and efficiently [375].

Other:

Several publications consider one or more phases of the SLA lifecycle as the focus of their contribution. A number of works focus on SLAs but their main contributions are classified as review, survey, or guide papers.

Hussain et al. [208] provide a comprehensive overview of the current state of the art related to Cloud-based SLA management approaches and their features

and shortcomings from the service provider's point of view when creating an applicable SLA.

Faniyi and Bahsoon [158] examine the SLA-based Cloud study landscape systematically to know the state of the art and to identify open issues. Considering an SLA for resource allocation, the results of the study show that: (1) a minimum number of SLA parameters are considered; (2) heuristics, policies, and optimisation are most frequently used for resource allocation methods; and (3) the style of monitor-analysis-plan-execute (MAPE) design predominates in autonomous Cloud systems. These findings contribute to the fundamental and autonomous management of engineering Cloud SLAs, as well as forming a motivation for further research and industrial-oriented proposals and solutions.

Moreover, as an SLA reflects an agreement in the context of a service provision between a Cloud provider and a Cloud-based service consumer, this raises the following question: how can we describe the SLA clauses between signatories, such as service rates, service quality constraints, penalties etc., in the case of an SLA breach? Maarouf et al. [292] present an extensive overview of how SLAs are created, managed, and used in the Cloud computing and web services paradigm. This study reviews a number of available works related to SLA language specifications. After that, a comparison of the reviewed studies is presented to highlight their strengths and weaknesses.

2.3.2 Works Related to SLA Applications Category

This category includes works that provide solutions, mechanisms, and policies to enhance the delivery of the required functionality while considering the SLA constraints at the same time. This category consists of the following subcategories: 1) Scheduling, 2) Load balancing, 3) Elasticity, 4) Resource provisioning and allocation, 5) Resource management, 6) Service placement, and 7)Orchestration. Table 2.3 maps the selected work to the the following sub-categories:

Table 2.3 Results of mapping published studies to SLA applications category

Scheduling	Load Balancing	Elasticity	Resource Provisioning/Allocation	Resource Management	Service Placement	Orchestration
[8], [261] [338], [29] [83], [273] [508], [518] [529], [527] [260], [112] [216] [223], [224] [452], [59] [242], [241] [381], [79]	[46], [115] [133], [171] [202], [359]	[353], [89] [536], [32] [40], [357] [236], [360] [441], [512] [42], [143]	[29], [453], [520], [150] [499], [345], [453], [159] [509], [286], [491], [496] [229], [413], [107], [500] [415], [105], [30], [442] [277], [513], [511], [492] [507], [352], [93], [48] [504], [535], [428], [177] [426], [506], [514], [437] [83], [317], [505],[230] [54], [231][329], [311] [139], [431]	[100], [313] [272], [109] [269], [478] [178], [92] [276], [402] [45], [271] [346], [101] [429], [528] [87], [2] [423]	[69], [310] [311], [44] [94], [431] [263], [458] [239],[139] [501], [459] [329] [468],[348] [279],[243]	[110], [409] [321], [39]

Scheduling:

This includes work that provides an SLA-aware scheduling technique. SLA-aware scheduling is a technique that considers SLA constraints, such as response time, when deciding which activity can start and which execution mode (e.g., space-shared/time-shared) can be applied. We have mapped 21 works to the SLA-aware scheduling sub-category. Among the mapped works, there are a number of works that consider SLAs in their proposed scheduling solution for Cloud services, such as: [8, 261, 338, 29, 83, 273, 529, 527, 508, 518]. For example, Leitner et al. [261] present a strategy to efficiently schedule incoming requests to virtual Cloud computing resources to minimise the sum of resource costs and SLA violations. However, the IoT is not the research focus within the above-listed works.

In order to achieve SLA-aware benefit optimisation in Cloud services, Moon et al. [338] present a resource scheduling analysis. In the performance evaluation phase, the results show that the choice of resource scheduling strategy based on resource scheduling analysis has a significant impact on the Cloud service provider's SLA delivery and overall gain. Furthermore, works such as [223, 224, 216] propose scheduling solutions for the Fog computing paradigm. Additionally, Stavrinides et al. [452] present a scheduling approach for real-time IoT workflows in both Fog and Cloud environments. Contrary to traditional approaches in which the primary processing of IoT tasks is carried out in the Fog layer, their approach aims to schedule compute-intensive tasks with low communication demands in the Cloud, and communication demand tasks with low computational requirements in the Fog. This makes use of potential differences in the schedule of virtual machines for the Fog and the Cloud. Nevertheless, because of the use of Cloud services, the proposed approach comes at significant monetary expense because it uses a reserved dedicated hosts [452] .

Load Balancing:

This includes works that provide SLA-aware load balancing techniques. Load balancing is an approach that allows the distribution of requests coming to a resource (e.g., VM) while taking into account the current load of that resource. For example, it aims to distribute requests among available nodes evenly. SLA-aware load balancing for the Cloud environment has been investigated by a

considerable number of studies, such as [46], [115], [133], [171]. For example, Ashouraei et al. [46] provide a parallel genetic algorithm-based approach for prioritising tasks. The aim is to use resources efficiently and reduce the waste of resources in Cloud environments. This is achieved by enhancing the load balancing rate when choosing better resources in a shorter time with a lower task failure rate to accomplish arrival tasks. However, the proposed algorithm is applied for load balancing purpose in the Cloud environment.

Additionally, a number of works have studied load balancing to improve the performance of the Fog environment under certain constraints. One example is [202], which integrates Fog, Cloud and software-defined networking to enhance the load balancing of the Internet of Vehicle. However, the main QoS metric that they consider is latency. Furthermore, Neto et al. [359] propose an algorithm to distribute the load efficiently in Fog environments. They consider resource utilisation, response time and multi-tenancy, but there is no consideration of other factors such as cost.

Elasticity:

This includes works that study the SLA-aware elasticity approach. Elasticity is defined by [323] as "Rapid elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time". We have mapped 12 works to this sub-category. Works such as [353, 89, 536, 32, 40, 357] study elasticity solutions for the Cloud environment to scale up or down under certain constraints. For example, [40] proposes a scaling model for the VMs that contains the services of the distributed applications, to scale dynamically based on variations in the number of users of the application. However, the proposed approach is applied only to Cloud infrastructure management.

El Kafhali and Salah [143] study elasticity for the Fog environment. They present a computational and analytical model to study and analyse the performance of the Fog computing system. The proposed mathematical model calculates the number of Fog nodes needed to satisfy the QoS parameters such as

response time, system loss rate, system throughput, and CPU utilisation, under any IoT workload provided. However, the effect of the approach on overall cost was not discussed.

Resource Provisioning and Allocation

This includes work that considers providing SLA-aware resource provisioning and/or resource allocation. The reason for combining both approaches (i.e., resource provisioning and allocation) is the lack of a clear definition that differentiates between the two. We mapped around 46 studies to this category. Resource provisioning and allocation concerns providing and allocating resources according to the workload, considering certain constraints. There are a number of studies that have contributed to the Cloud environment, such as [29, 453, 499, 345, 509, 286, 229, 413, 453, 159].

For example, work in [29] models user requests considering budget and deadline constraints. It then models infrastructure resources as a list of data centres, VMs, data sources and network throughputs. It proposes cost-aware and SLA-based algorithms to provide Cloud resources and schedule analytic tasks. Moreover, the research presented in [345] allows developers to automate resource provisioning. It allows developers to specify their preferences for resources and resource attributes, then, the system can propose solutions that match their needs. However, in reality, resources in the Cloud have a limit, thus when consumers submit a massive number of requests, a Cloud provider may need to lease resources from other providers. Therefore, the study in [105] proposes a combinatorial auction-based approach to dynamically provide and allocate resources to solve this problem, considering consumer time limit constraints. However, all of these works are applied to allocate resources in the Cloud environment.

There are a number of works for the Fog/Edge environment, such as: [277, 513, 507, 352, 329, 311, 139, 431, 505]. Yao and Ansari [506] research the trade-in maximising reliability and minimising system costs for the provision of Fog resources in IoT networks. They formulate an integer linear programming problem as an algorithm to address the aforementioned issue but it suffers from high computational complexity. They then propose an alternative approach with

better time efficiency to deliver suboptimal solutions. However, the considered QoS requirements are not mentioned explicitly, except within the evaluation. They consider response time as a QoS requirement to reflect the fact that the approach is capable of trading-off between cost and service delivery within the predefined time constraint.

Resource Management:

This includes works that provide SLA-aware resource management. Resource management performs more than one task including, but not limited to, scheduling, selection, load balancing, resource provisioning, and orchestration. We have mapped around 19 studies. Works such as [272, 109, 269, 478, 178, 92] propose SLA-aware management solutions for the Cloud environment. For example, [109] provides an SLA-aware solution that can provide green resource management for the Cloud infrastructure. Furthermore, [402] proposes a novel learning automata-based algorithm that enhances the use of resources and decreases energy consumption. The proposed algorithm takes into account changes in the resources required by the user to predict the Physical Machine (PM) that may be overloaded. Because the proposed algorithm avoids database congestion, it increases the use of PMs, reduces the number of migrations and shuts down idle servers to minimise the data centre's energy consumption. However, the proposed approach is not applied to the IoT.

Serrano et al. [423] illustrate a Cloud service management using matchmaking operations and applying self-management principles. These enhance the distribution and management of IoT services among various Cloud providers and they use the analytical results as a mechanism for controlling applications and the deployment of services in Cloud systems. They claim that the proposed approach can be applied to Cloud-based IoT applications or Cloud systems; however, they applied it to manage IaaS in the Cloud computing domain.

Service Placement:

This includes studies that consider consumers' constraints when mapping the required services to a resource in order to accomplish that required service[6]. We have mapped 17 works to the SLA-aware service placement approach sub-category. Most available works propose placement policies to map tasks to Fog resources such as [329, 431] and/or across Fog and Cloud resources such as [310, 69].

Mahmud et al. [310] propose a profit-aware application placement strategy for integrated Fog-Cloud systems to address these issues. It is implemented using the Integer Linear Programming Model, which simultaneously improves profit and guarantees QoS during the process of application placement. It also provides customers with compensation for any violation of their SLA. In a simulated Fog-Cloud environment using iFogSim, the quality of the proposed policy is tested and the results show that there is an increase in the profit level of the provider and the satisfaction rate of the consumer. However, application constraints such as deadline is considered at the application level only. No consideration is given to deadline constraints for each involved activity that delivers the application, since it considers placing the application as a whole on one instance without considering the possibility of having sub-modules that can be distributed across layers.

Ben et al. [69] introduce Network Function Virtualisation (NFV) placing and optimisation approaches across the Edge and Cloud carrier network. They take into account QoS constraints and the use of queuing and QoS models. The main design objectives are to maximise resource utilisation, prevent cloudlet overload, and avoid violations of the SLA. Through extensive simulations, they show how these conflicting goals can achieve a trade-off. Skarlet et al. [431] investigate the placement of IoT services across Fog instances, taking their QoS requirements into consideration. They demonstrate that the proposed optimisation model avoids QoS violations and reduces the execution costs by 35 percent compared to a Cloud-only approach. In maximising the use of the Fog environment, the application QoS metrics, i.e. deadlines for applications, are taken into account.

[6]Due to the fact that we are interested in service/activity placement, this section is more detailed than others

However, budget constraints are left as future work.

The work in [311] details the evolution in the design and development of the Fog computing architecture for IoT services. It optimises the number of Fog resources in order to decrease the total latency generated by aggregation and processing. The results show that an optimum deployment of Fog nodes can reduce latency in comparison to a traditional Cloud environment. However, this work is architecture oriented and it focuses on the number of deployed Fog nodes rather than mapping services to Fog nodes. Apat et al. [44] implement an efficient architecture for managing an IoT system application for Fog layer service and an analytical model to determine the placement of services and energy consumption in IoT-Fog-Cloud scenarios. As they focus on reducing energy consumption, it is necessary to check the power consumption of a device before using it; thus there is a need to compute the working hours of devices and their idle time during the service request. They formulate an energy equation, apply various optimisation techniques, and compare the quality of the proposed techniques with other work. However, this work is mainly concerned with energy consumption.

Kochovski et al. [239] introduce a new decision-making approach with an optimal QoS for database containers. They also provide software engineers with QoS guarantees. Lastly, a multi-stage orchestrating approach is provided to automate the entire process of using Big Data applications to automate. QoS measurements from a distributed monitoring system are the input for the proposed Markov Decision Processes method. The measurements, obtained with QoS constraints, are used later to derive models for particular workloads and database deployment. The created models are automatically produced. However, the main focus of this work is related to database container placement.

Taneja and Davy [459] present a module mapping algorithm to utilise Fog and Cloud infrastructures for IoT applications. The calculation is dynamically spread through Fog and Cloud layers, and the modules can be deployed on Fog layer devices close to the source. The proposed algorithm is generic and it can be applied to different network typologies for a wide range of standardised IoT applications, regardless of the workload. However, this work only considers mapping modules based on their computing requirements and it finds resources

with an appropriate processing capacity, whereas we aim to consider budget, the deadline associated with each task, and computing capacity constraints.

In [329], referring to a multi-layer Fog computing architecture for IoT service provisioning, a new mechanism for service placement is proposed to optimise the decentralisation of services in the Fog computing environment. This is achieved by leveraging context-aware information such as location, QoS, and time. An experiment is conducted with several models of smart grid applications in order to test the proposed approach. The results show that the proposed approach is effective when compared to Cloud-only models with regard to reducing latency, power consumption, and networking loads. However, the proposed work does not consider the deadline constraints of each of the applications' modules.

Tran et al. [468] offer a new, multi-layer, IoT-based Fog computing architecture. In particular, they develop a service placement mechanism that optimises service decentralisation in the Fog landscape by using context-aware information such as location, response time, and service resource consumption. The approach is used in an optimal way to increase the efficiency of IoT services in terms of response time, energy, and cost reduction. Experimental results from simulated data and real-world applications show the efficiency of the solution. It optimises Fog device use and reduces latency, energy consumption, network load, and operating costs. The results show that the proposed system is robust and that it is capable of maximising IoT potential. However, this work considers the tasks to be independent while in reality, IoT applications' modules are not independent.

Naas et al. [348] present the iFogStor approach, which aims to reduce the overall latency of storage and data retrieval in Fog. They formulate the data placement problem as a GAP (Generalised Assignment Problem) and propose two solutions: 1) an exact solution using integer programming; and 2) a geographically based solution to decrease the solving time. Both solutions are proven to be very good, as latency is lowered by more than 86% in comparison with a Cloud-based approach and 60% in comparison with a naive Fog solution. The use of the heuristic geographical zoning process can effectively solve problems with many Fog resources and make iFogStor possible and scalable in a few seconds. How-

ever, its focus is related to storing data at the Edge in order to ease data retrieval.

IoT service placement in Fog architecture is studied in [139]. The authors propose an infrastructure and IoT application model as well as a placement approach, taking into account the power consumption of a system and minimising delay violations using a Discrete Particles Swarm Optimisation (DPSO) algorithm. iFogSim simulator is used to evaluate the proposed approach. The results are compared with: Binary particle swarm optimization (BPSO), Dicothomous Module Mapping (DCT), IoTFogOnly, IoTCloud (IC) and Fog-Cloud (FC) placement approaches. However, it only considers the effect on energy consumption and delays, while we aim to reflect the impact on cost and time, in addition to energy consumption.

In [279], the authors apply queuing theory to carry out a comprehensive study on energy use, delays in execution, and the costs of the offloading process in a Fog computing system. Three queuing models are used in mobile devices and Fog and Cloud centres, and data rates and wireless connection power consumption are explicitly considered. The theoretical analysis formulates a multi-objective optimisation problem with a common aim of minimising energy consumption, execution delay, and payment costs by selecting the optimal probability for offloading and transmitting power for each mobile device. Extensive simulation studies are performed to show the effectiveness and performance of the proposed scheme compared to several existing schemes.

Kolomvatsos and Anagnostopoulos [243] suggest a smart decision-making system to assign tasks locally. The remaining tasks in the network or the Fog/Cloud will be transferred to peer nodes. To minimise the execution time, they implement a two-step decision process. The first step is to decide whether or not a task can be performed locally; if not, the second step includes the advanced selection of the most suitable peer to assign the task to. When no node is capable of performing the job throughout the Edge network, it is then sent to the Fog/Cloud for maximum latency. They assess the suggested system thoroughly, showing its applicability and optimally on the Edge of the network. However, their view is that tasks should be processed in a sequential order, which is not typically the case with an IoT application, where there is the possibility that tasks can be run

in parallel.

Orchestration:

This includes works that contribute to orchestration while respecting SLA constraints, such as [110, 409, 321, 39]. Chhetri et al. [110] define Cloud resource orchestration as "The process of provisioning computing resources comprises the following phases – selection, assembly and deployment of computing resources, and monitoring of deployed resources [400]".

In [409] the authors introduce e-eco, an energy-efficient Cloud orchestrater that enhances the trade-off between power savings and application efficiency through a series of power-saving techniques. A prototype is developed and tested in real and simulated Cloud environments, and the tests show that e-eco preserves the balance between power savings with minimal impact on performance.

From the IoT perspective, Mechalikh et al. [321] propose a task orchestration for the IoT. They focus on Edge computing's role in ensuring a high scalability environment. The study introduces an algorithm for task orchestration based on the Fuzzy Decision Tree. It leverages learning from reinforcement, which helps it to respond to unexpected changes in the environment. The proposed design offers greater scalability and low delays, regardless of the number of devices, compared to existing solutions. The approach considers QoS requirements such as CPU utilisation, delay and energy consumption. However, there was no mention of the impact of the proposed approach on the cost of the system when applying the proposed task orchestration.

2.4 Discussion

This section describes the outcomes of the study and responds to the research questions identified in Section 2.2.1.

RQ1. What are the current research topics related to SLAs for the IoT?

The main focus of our research falls into the first category, which consists of

technical works related to the SLA lifecycle, in particular SLA specification. Table 2.2 maps the retrieved works' topics to the most technical approaches related to the SLA lifecycle. These include SLA specification, negotiating, monitoring enforcement, and management. Figure 2.7 depicts the mapped publications to each sub-category, revealing that monitoring and negotiation are among the most discussed topics.

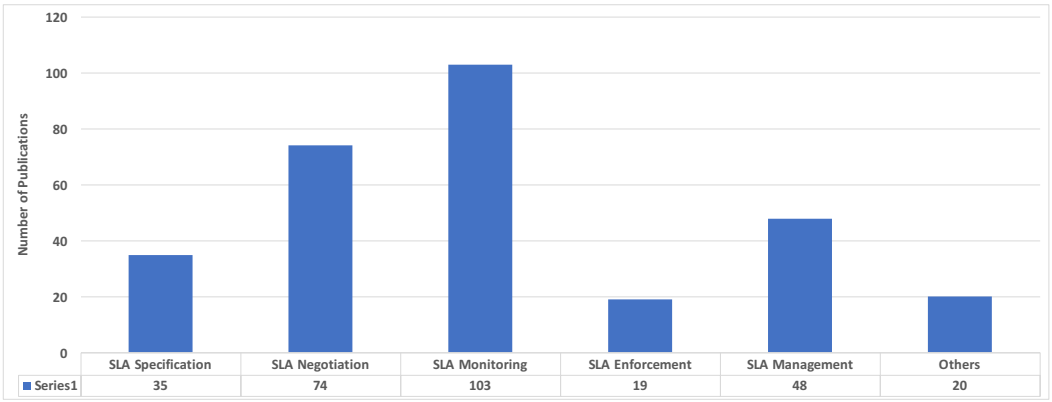

Fig. 2.7 Result of mapping relevant publications to the subcategories of SLA lifecycle category

For the second category, which is SLA applications, we consider only work which states clearly that the main focus is SLA oriented. Most of the retrieved works are related to the Cloud, i.e. there is little work related to the Edge, Fog and IoT paradigms. Thus, we consider QoS-oriented studies to find works performed for the Fog, Edge, IoT environments. Table 2.3 reflects the results of this systematic mapping study by mapping the retrieved works' topics to the most SLA-aware approaches. These include SLA-aware scheduling, load balancing, elasticity, resource provisioning/allocation, and resource management. Figure 2.8 depicts the mapped publications to each sub-category, showing that resource allocation, management, and scheduling are among the most investigated topics.

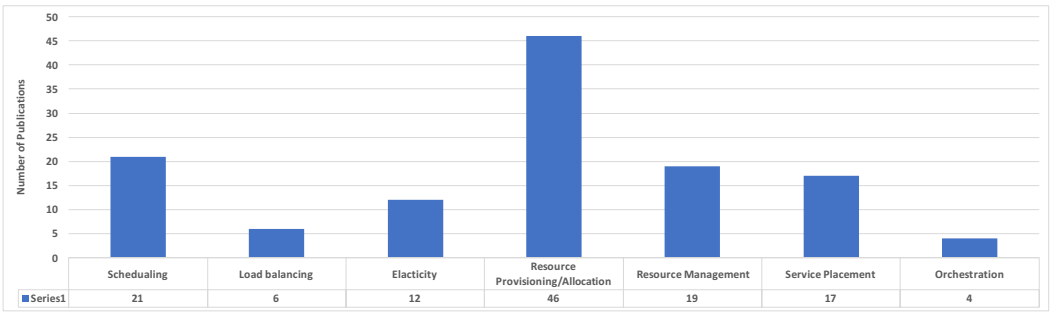

Fig. 2.8 Result of mapping relevant publications to subcategories of SLA-aware approaches category

RQ2. How active are the research topics related to SLAs? (Measured by capturing the number of publications in the last decade)
There are a considerable number of works that have been published within the last decade. Figure 2.9 shows the number of publications per year related to the SLA lifecycle. Years 2013 and 2014 are among those with the highest number of publications, which reflects the association with the period of the Cloud's growth, especially given that most of the published works are for the Cloud environment.

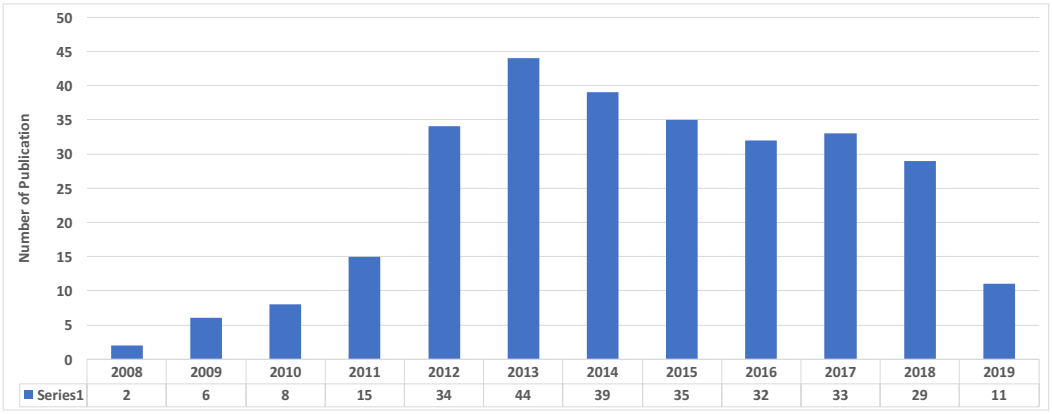

Fig. 2.9 Result of year-based classification of the relevant publications related to the SLA lifecycle category

Figure 2.10 shows the number of publications per year under the SLA applications category. Years 2016, 2018 and 2019 are among those with the highest number of publications. However, most of the mapped publications that contribute to the Fog/Edge paradigm were published in 2018, beside works on the Cloud paradigm, which might explain the high level of published works in this year.

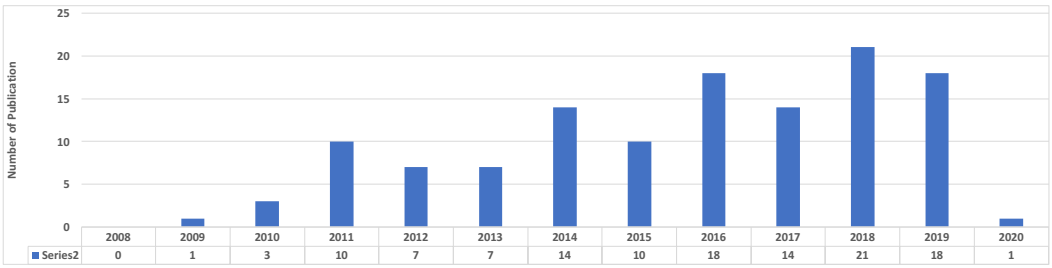

Fig. 2.10 Result of year-based classification of the relevant publications under the SLA-aware approaches category

RQ3. What is the number of publications per year related to SLA specifications?

The number of relevant publications per year in the research area related to SLA specifications is plotted in Figure 2.11. We are interested in work related to SLA specifications for the IoT. However, the work presented in Figure 2.11 is related, mostly, to the Cloud. Some are specifically for one of the Cloud layers such as CSLA or for all the Cloud layers such as SLAC [245, 476].

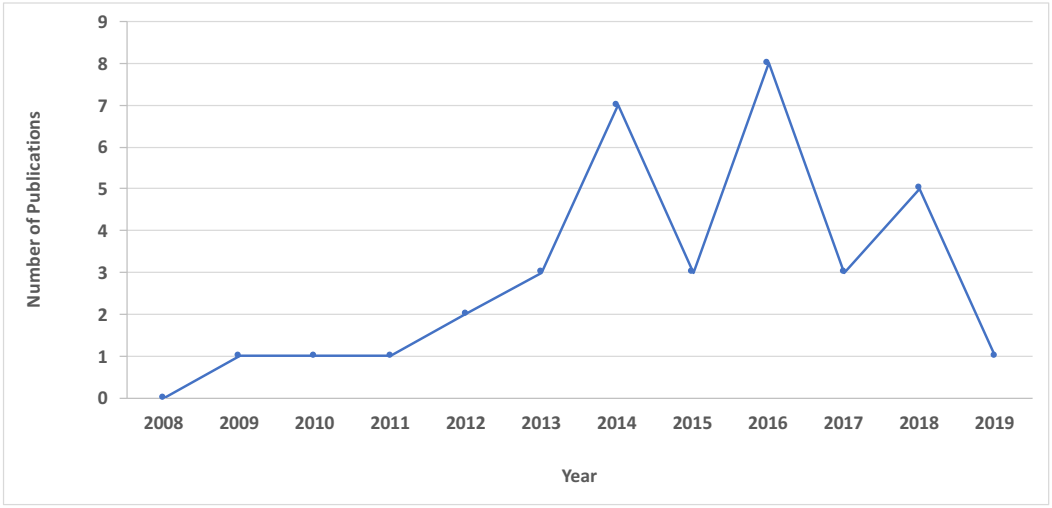

Fig. 2.11 Number of Publications Related to SLA Specifications per year

RQ4. What is the number of publications per year related to SLA-aware service placement?

The number of relevant publications per year related to SLA-aware service placement is plotted in Figure 2.12. Years 2018 and 2019 reflect the highest number of publications.

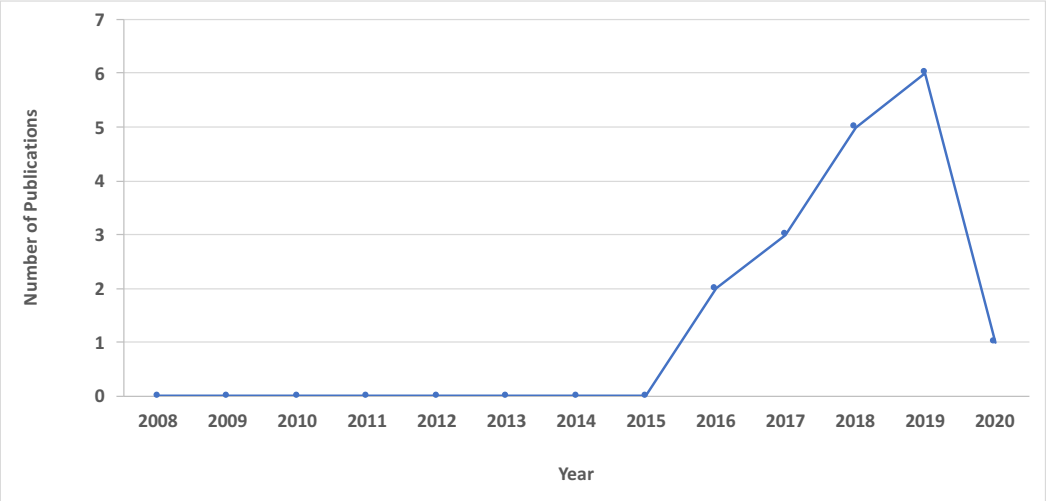

Fig. 2.12 Number of Publication Related to SLA-aware service placement. Specification per year

RQ5. What are the research gaps that need to be addressed in future studies?

From this systematic mapping study, we have identified a number of research gaps derived from the literature insight provided in Section 4.2. These gaps can be covered by future research.

- The first gap is a lack of studies on standardising SLA specifications/definitions for the IoT. In particular, there are few studies that consider the multi-layered nature of the IoT, where there are several different software and hardware components across layers (IoT devices, Edge computing, Cloud computing). Therefore, many contractual parties (e.g., Cloud provider, Networking provider) need to cooperate, which increases the need to unify the vocabularies used in order to reduce ambiguity.

  Although there is a considerable number of works related to SLA specifications/languages for Grid Computing, web services, and the Cloud, from the IoT perspective there is a lack of research in this area (as reflected in the collected references that are listed in Table 2.2). As a result, according to [375], future factories would need SLA standardisation, with the possibility of handling all the aspects addressed autonomously and effectively. It needs to handle aspects from the specification to the negotiation, from monitoring

to enforcing SLAs [375].

According to [472], the extensions needed to support the domain. The availability of domain-specific vocabularies are a significant feature in an SLA definition language [472]. A domain SLA language must therefore not only be straightforward for all the involved parties but also support the domain [472]. Furthermore, to avoid ambiguity in SLAs and disagreements over the meaning of contracts, the SLA vocabularies must have formal interpretations [472]. Moreover, providing a machine-readable end-to-end SLA for an IoT application will enhance SLA management since a machine-readable SLA can automate the process of selecting a service provider and negotiating, monitoring, enforcing, and managing the SLA [472, 290].

- Current SLA negotiation literature is limited, especially when considering large-scale, dynamic environments such as the IoT. The negotiation protocol is a crucial area of the SLA negotiation phase and is discussed in various Cloud projects. However, existing IoT negotiation strategies follow a centralised approach, which may not be realistic, given the dynamicity and distributed nature of the IoT environment [267]. However, Li et al. [267] suggests a negotiation process using decentralised network brokers to negotiate efficiently on behalf of service consumers with multiple IoT service providers. The framework uses a hierarchical architecture to manage the message flows during the negotiation process and to cluster service information. Nevertheless, there is a need for further research to consider an efficient, dynamic negotiation protocol, since SLA negotiation barely discussed in IoT contexts [265]. For example, it would be valuable to propose a mechanism to calculate an acceptable time limit and finalise the negotiation, considering the sensitivity of IoT applications.

- The nature of IoT interaction means that different components need to interact with each other and those components might be provided by different providers. Each one may have its SLA which specifies its QoS capabilities. As a result, one of the big challenges is how to collect and integrate the metrics among those different providers for monitoring purpose.

  However, from the IoT perspective, there is still a need for an end-to-end SLA monitoring. It enables organisations to ascertain the cause of any

performance issue they are experiencing, whether it be the application design, the infrastructure of the network, or the Cloud service provider [408].

- There are a considerable number of studies in the literature covering SLA management. However, there is a shortage of available feasible SLA management frameworks that can standardise the SLAs with the possibility of managing all aspects: definition, negotiation, monitoring and enforcement of SLA for IoT, autonomously and efficiently [375]. Furthermore, available SLA frameworks float between being too specific or too generic [165], thus, we believe that there is a need to provide an SLA management framework to address the general requirements of the IoT applications as well as consider domain-specific requirements.
- Utilising Blockchain-based smart contracts for SLA monitoring, enforcement, and compensation is in its early stages [418]. There are a few works that investigate and apply Blockchain-based smart contracts for SLAs such as [33, 418, 533]. However, most of the available works are in their early stages and are mostly tested using prototypes or simulation environments. Thus, there is a clear research gap in investigating the application of Blockchain-based smart contracts in association with SLA lifecycle phases for the IoT paradigm. It can be applied to enforce SLAs and overcome the trust issues concerning the monitoring of SLA violation and compensation.

Generally speaking, most available works are for the Cloud environment, however, in the last three years, there has been considerable work covering the Edge/Fog/IoT environment. While some of the work investigated SLAs or QoS related to Cloud, Edge/Fog, and/or IoT paradigms, considering SLAs across these layers still requires further contributions. Nevertheless, one of the key reasons for having research gaps is that the topic is still young and the scientific community is still developing SLA management frameworks, languages, and techniques [375].

2.5 Conclusion

This chapter provided a background overview of the research undertaken in SLAs in the last decade. As many IoT applications depend on Cloud and Edge resources for computing, analysing, and storage purposes, this chapter presented brief knowledge related to Cloud computing and Edge computing. Additionally, it explored previous works relevant to SLAs in general and to the SLA lifecycle in particular. From the IoT perspective, there is a lack of research focusing on SLA specification, negotiation, monitoring, enforcement, and management; most of the presented work is Cloud based.

In this research, our interest is mainly related to SLA specifications for the IoT and we consider the multi-layered nature of the IoT since a number of contractual parties (e.g., Cloud provider, Networking provider) need to cooperate according to user requirement constraints, as specified within the SLA. Therefore, an SLA specification language should consider that consumers and parties are not domain experts; thus, the language for the description of SLAs should be easy to understand [472]. Also, an essential feature of an SLA language is the possibility to extend it to provide and support the domain-specific vocabularies [472]. Therefore, a domain SLA language must not only be straightforward for all the parties involved but it must also provide domain-specific support features [472]. Furthermore, to avoid ambiguity in SLAs and disagreements over the meaning of contracts, the SLA vocabularies must have formal interpretations [472]. Therefore, in the next chapter, we propose an IoT conceptual model as well as rich domain-specific vocabularies as a first step towards proposing an end-to-end SLA for the IoT.

Chapter 3

SLA Conceptual Model for IoT Applications

Overview

Since SLAs specify the contractual terms that are formally used between consumers and providers, there is a need to aggregate the QoS requirements from the perspectives of Clouds, networks and devices to deliver the promised IoT functionalities. Therefore, the main objective of this chapter is to provide a conceptual model of an SLA for the IoT as well as rich vocabularies to describe the QoS and domain-specific configuration parameters of the IoT on an end-to-end basis. We first propose a conceptual model which identifies the main concepts that play a role in specifying end-to-end SLAs. Then, we identify some of the most common QoS metrics and configuration parameters related to each concept.
We evaluate the proposed conceptual model using a goal-oriented approach. The participants in the study report a high level of satisfaction regarding the proposed conceptual model and its ability to capture main concepts in a general way.

3.1 Introduction

IoT applications are mostly time-sensitive applications. Thus, it is important to consider when data need to be collected, what the next processing step is, and where to process each step. Therefore, there is a need to aggregate QoS requirements from the perspectives of Clouds, networks and IoT device layers to deliver

the promised IoT functionalities at the required quality level, as agreed upon within the SLA. Furthermore, the associated QoS requirements for each step should be specified in an unambiguous way by formally defining them, so their meaning and used terminology are known and unified. For further illustration, consider the following use-case scenario:

3.1.1 Remote Health Monitoring Service (RHMS)

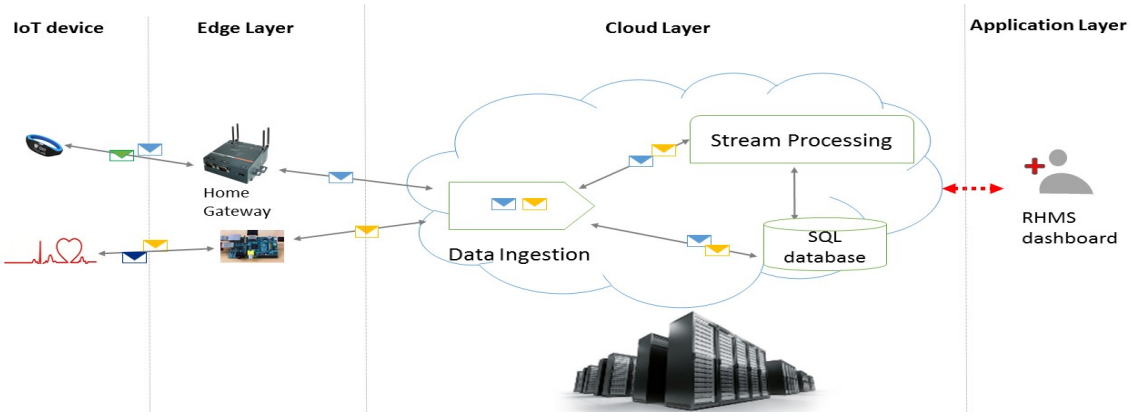

Fig. 3.1 cooperated layers to deliver RHMS

In a remote health monitoring service (RHMS), patients are monitored remotely, and if there is an urgent case, it can be detected and even predicted to avoid health-related disasters (Figure 3.1 shows the typical layers involved in an RHMS). An RHMS uses body sensors, radio frequency identification readers and accelerometers to monitor changes in heart rate and sugar levels for the timely administration of medications and the avoidance of falls. An RHMS involves many workflow activities, such as collecting real-time data from various IoT devices (e.g., sensors to capture the patterns of human activity and video-streaming cameras). The collected data are analysed to determine whether there are any abnormal activity patterns (e.g., the possibility of a heart attack) by comparing newly collected data with historical/stored data. The users of this application will require real-time constraints. For example, events such as heart attacks or falls should be detected within milliseconds, and automatic alerts should be sent to emergency services within seconds of detection. To achieve these high-level objectives, a list of nested objective constraints should be used to ensure that

the service is provided to the subscribed patients at the same level as they expect.

This sort of application is time-sensitive, meaning that any unforeseen delay in one or more phases of the data flow (e.g., collection, transfer, ingestion or analysis) will affect the accuracy and appropriateness of the actions taken. For example, a delay in the network could lead to a late response, which could cause harm to the subscribed patient. Thus, the application's performance depends not only on the provided functionality but also on the quality of the services offered across Edge and/or Cloud computing environments, which can be affected by resource capabilities and other configuration parameters. As a result, the specifications of low-level technical requirements need to be considered due to their extreme impacts on meeting the QoS requirements of an application. In RHMS, in addition to the constraints at application level (e.g., an end-to-end response time less than X time units), more than one constraint must be considered for each involved service. For instance, the data collected from IoT devices should be accurate and up to date. Moreover, it is important to minimise the latency of data pre-processing (e.g., data filtering), which can be achieved using a raspberry pi, mobile phone or Edge server, among other options. Furthermore, it may be possible to perform an analysis and compare incoming data with historical data or the results of a predefined model. Consequently, more than one constraint must be considered: applying machine-learning algorithms requires a high accuracy constraint, stream processing requires a low latency constraint and batch processing requires a high throughput constraint.

As a result, we attempt to contribute to the SLA of the IoT by proposing a conceptual model that captures the knowledge base of IoT-specific SLAs. **This chapter contributes to the SLA for the IoT by:**

- Proposing an SLA conceptual model.
- Introducing key concepts of SLAs for the IoT and the related vocabulary terms that can be used for specifying QoS and configuration parameters;
- Evaluating the proposed conceptual model using a questionnaire-oriented approach from the domain experts' point of view.

In the following text, Section 3.3 introduces the proposed conceptual model for IoT applications. Then, Section 3.4 presents the vocabulary terms that can be

part of an SLA to reflect the QoS and configuration requirements. We evaluate the proposed conceptual model in Section 3.5.

3.2 Related Work

With the ever-increasing number of service providers and solutions for IoT-based services, there is rising demand for a mechanism that will regulate and automate the transactions and activities between the parties involved. This mechanism needs to take the form of an SLA. Although SLAs exist, we believe that in their current format they are unable to accommodate the unique characteristics of IoT applications. This section therefore provides details of a number of commonly used SLAS.

In web services, there are two main specification languages for the SLA. In the following we discuss the main components involved within WSLA and WS-Agreements.

WSLA [228] reflects all the details usually found in the SLA agreement [378, 293]. This information is based on three main parts (see Figure 3.2): 1) Parties: where service consumers and service providers are called the Signatory Parties and other external agents such as third parties are called the Supporting Parties. The signatory party description includes identification and technical properties such as address and how they accept events, while supporting parties have additional attributes that indicate the sponsor of the supporting party. 2) Service Description: to specify the features of the service and its parameters. 3) Obligations: to define the guarantees and constraints of the SLA parameters.

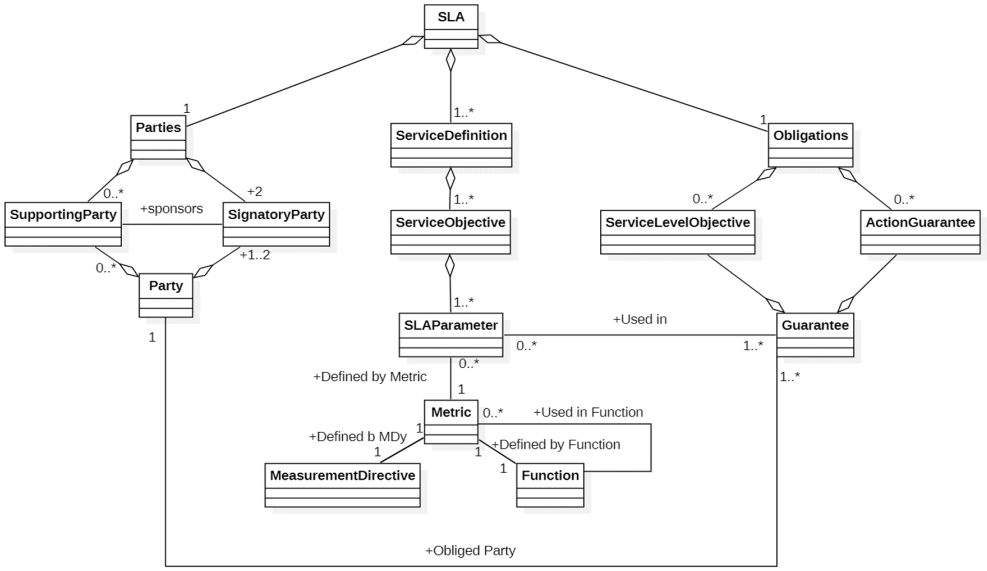

Fig. 3.2 WSLA conceptual model [378, 293]

The **WS-Agreement** is formulated as follows in three parts[293]: 1)The Agreement Scheme (Agreement Development Offer Scheme). This is used by the Agreement Initiator to create a template-based offer. The offer and the agreement are structurally the same. The agreement offer includes 1) the agreement name, the context (including the parties involved and the life span of the agreement), and the terms, which are the most critical aspect of an agreement offer. each term has at least one service term and zero or more guarantee terms that could be merged using logical operators. 2) Agreement Template Schema: The Agreement Respondent uses this template to promote acceptable offers. 3) Port type and operations: these are used to coordinate and control the life-cycle operations of the agreement (see Figure 3.3).

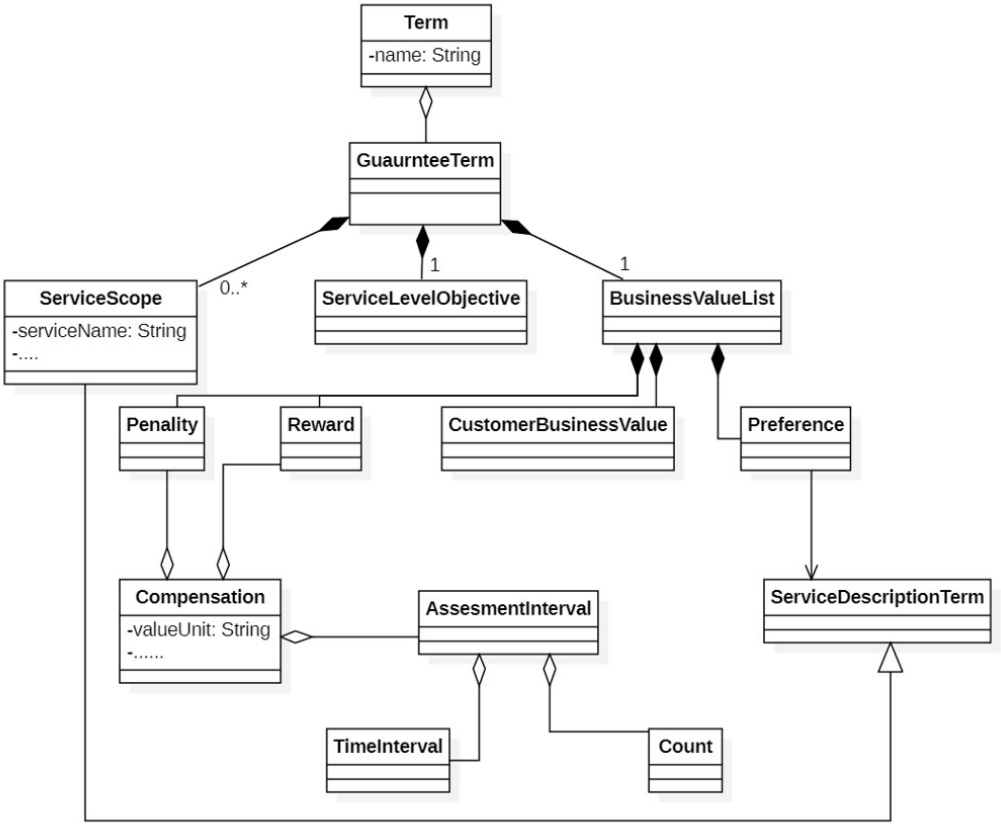

Fig. 3.3 WS-Agreement conceptual model [378, 293]

In Cloud computing, there are number of specification languages for the SLA such as SLA* and CSLA. In the following we discuss the main components involved within SLA*, CSLA and SLAC.

SLA* [227] has an SLA Template (SLA(T)) that comprises five parts (see Figure 3.4):1) SLA attributes template SLA; 2) the parties to the agreement; 3) service descriptions; 4) variable declarations; and 5) the terms of the agreement. The parties are defined by their position in the abstract SLA(T) syntax (supplier, customer). The service description is described by the interface statements. A declaration interface is used to associate a local interface with an identifier, and the local interface can be a functional interface or a resource description. To improve readability and avoid repetition of text, variable declarations are provided. The clauses of the agreement are formalised as two-type guarantees: action warranty and status. In addition, the SLA proposes a formal model for formalising penalties.

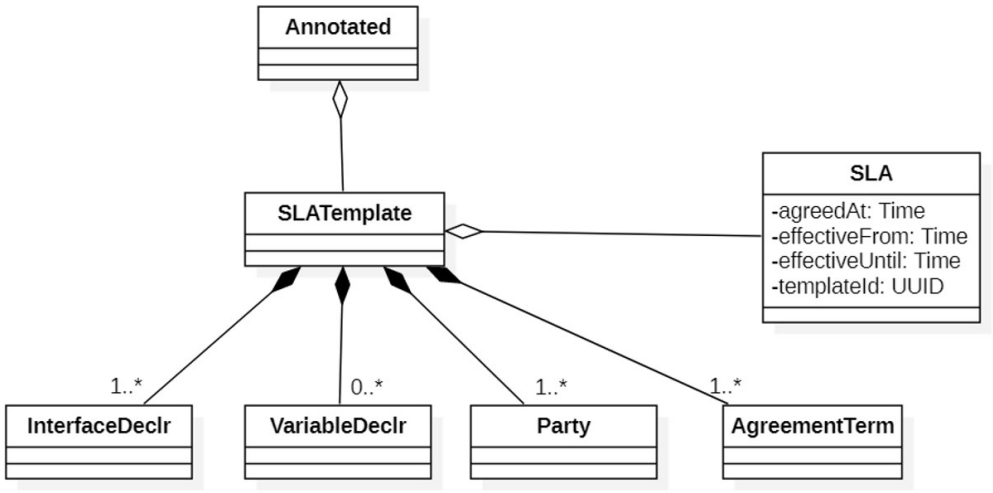

Fig. 3.4 WS-Agreement conceptual model [378, 293]

A CSLA [245] language comprises three parts (see Figure 3.5): a section that specifies the scope of the agreement, a section that describes the parties to the agreement and a section that refers to a template for the agreement. The validity determines the length of a contract. A CSLA considers two groups of parties: the signatory parties, which include service providers and consumers

of services, and supporting parties (for example, trusted third parties). A CSLA template is similar to an SLA pattern and it consists of the following elements: Definition of cloud services, Parameters, Guarantees and Terminations. Any XaaS service (SaaS, PaaS or IaaS) is specified in a Cloud service specification. The authors in [245]) suggest the concept of functionality degradation (i.e. standard vs. degraded mode) for each SaaS and PaaS application to handle any unpredictable and dynamic environment (i.e. 3D vs. 2D display). Parameters provide a means of identifying variables within the agreement context. Guarantees provide a range of guarantees. Every guarantee is composed of four elements: scope, requirements, terms and penalties. CSLA offers two billing types: a Pay as You Go plan and an All-in plan. Ultimately, the agreement starts based on the time specified within effectiveFrom and it ends before the time specified within effectiveUntil, which are defined within the Termination section.

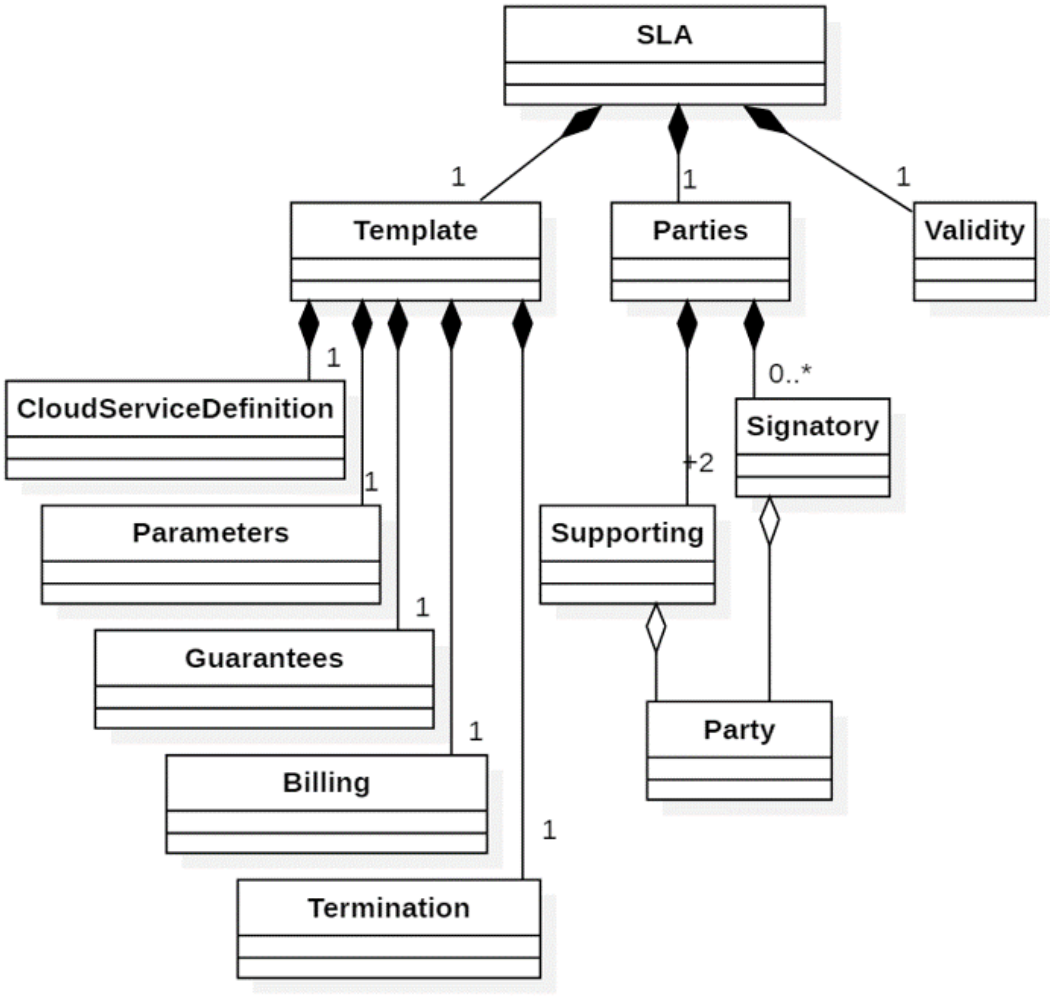

Fig. 3.5 WS-Agreement conceptual model [293]

SLAC is [476] defined as a domain-specific language for the Cloud. In SLAC, the key elements of an SLA are: the contract description, the specification of terms and the definition of the guarantees provided for those terms. Figure 3.6 presents the main elements of the conceptual model of a term definition in the SLAC.

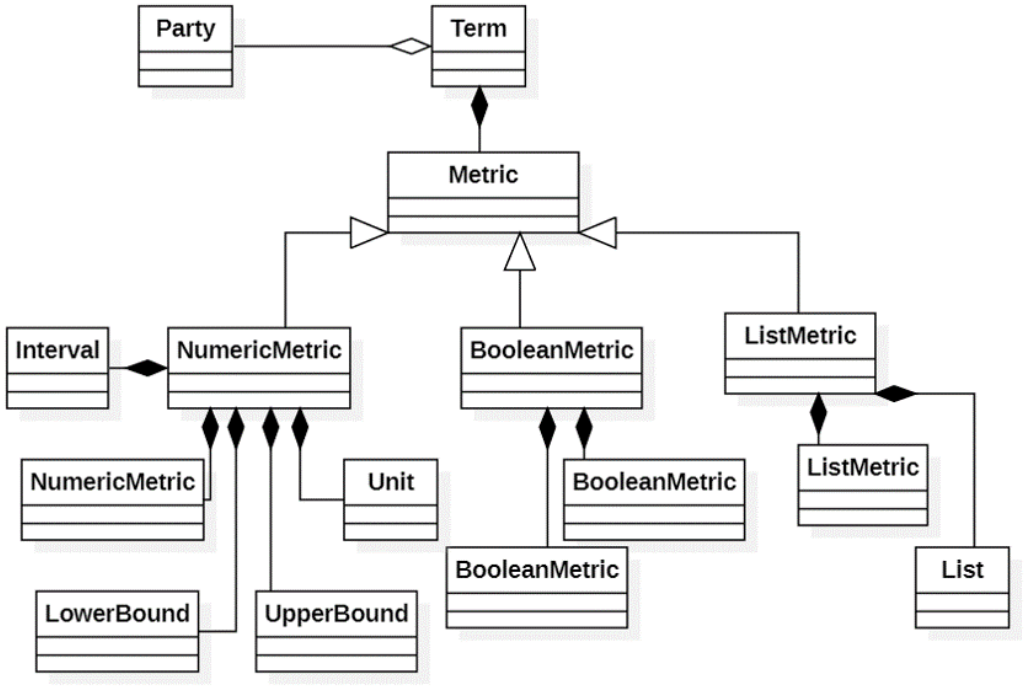

Fig. 3.6 The definition of a term in the SLA. The term is used to define a metric in SLAC [293]

In summary, our work varies considerably from those mentioned above: it emphasises the formal aspects of domain-specific SLAs by considering the particularities of the IoT domain, and it provides a base for defining an SLA on an end-to-end basis. Due to the multi-layer nature of the IoT and the importance of considering constraints on data while they flow across layers, we introduce the "workflow" concept. The reason behind defining the "workflow" concept is to allow us to define the associated requirements for the involved components on both the hardware and software levels. The next section introduces our proposed SLA conceptual model for IoT.

3.3 An End-to-End SLA Conceptual Model for IoT Applications

An end-to-end IoT ecosystem includes components through which application data flows. Components can include services (e.g., a sensing service or a real-time analysis service), infrastructure resources (e.g., IoT devices, Edge resources and Cloud resources) and/or humans. In an end-to-end SLA, it is important to consider the requirements of all of the services and infrastructure resources involved in delivering the IoT application. Considering an SLA on an end-to-end basis is essential because the level of quality at which the involved services are delivered has an impact on the SLOs at the application level. For example, in an RHMS, an SLO (SLO_{app1}) for urgent case detection, which requires a response within less than Y time units, is an SLO at the application level, and it involves many activities, such as analysing real-time data. Analysing real-time data requires a stream-processing service at an acceptable level of latency, and if the stream-processing service exceeds this level, then the SLO_{app1} at the application level might be violated.

As a result, we propose a conceptual model that captures the knowledge base of IoT-specific SLAs. The conceptual model expresses the key entities of the IoT ecosystem and the relationships between those entities within the SLA context. Due to the lack of a standard IoT architecture, we refer to the reference IoT architecture as presented in (Section 2.1) to identify the main concepts and the relationships between them. Additionally, since we aim to capture SLA requirements on an end-to-end basis, we define a new concept, "workflow activity", within the conceptual model. The purpose of this is to allow us to specify requirements related to services and infrastructures where data is flowing in between. This led us to associate "Workflow activity" with the "service" and "infrastructure resource" concepts. Other concepts: SLA, SLO and Party, are derived from previous works discussed above such as [228].

Figure 3.7 presents our conceptual model. In the following section, we describe the concepts covered in the conceptual model and give a brief discussion of the relationships associated with these concepts.

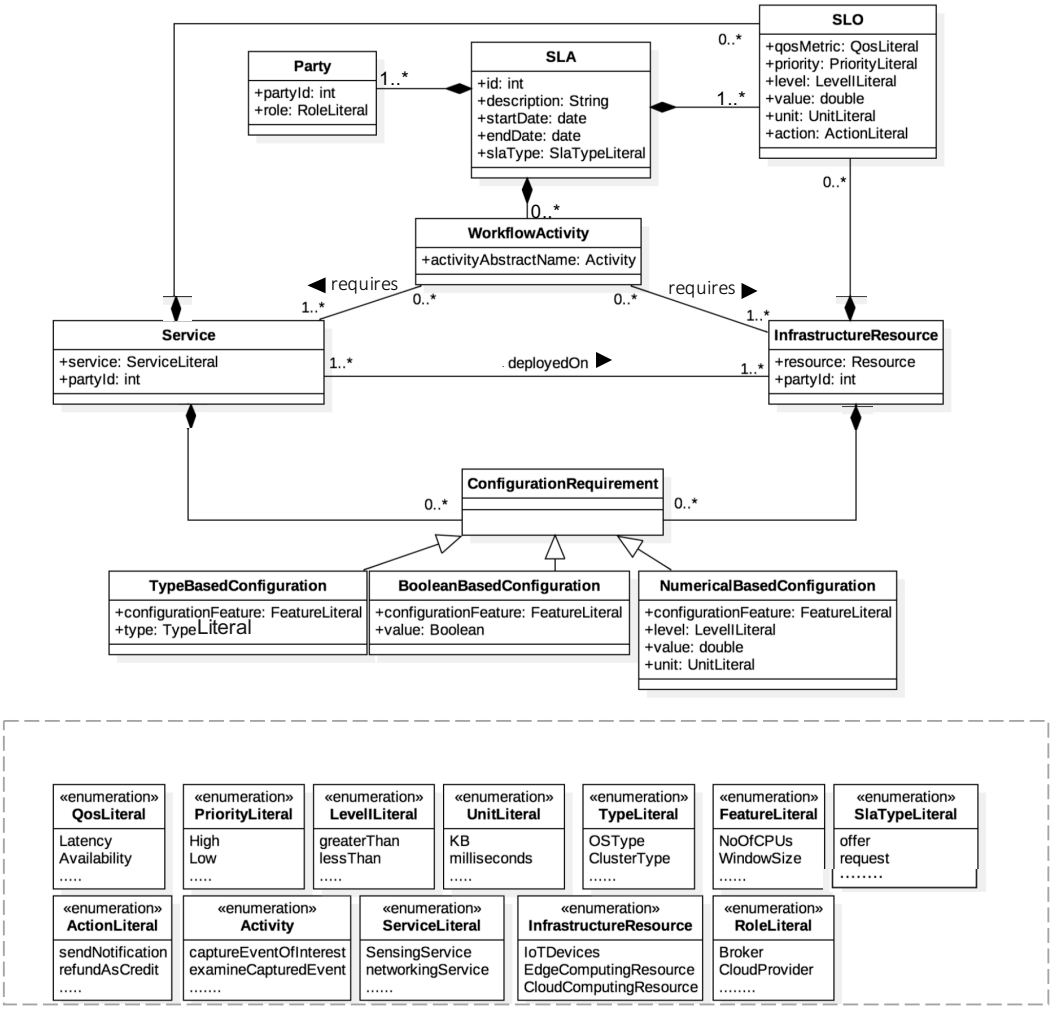

Fig. 3.7 SLA conceptual model for IoT applications that captures the key entities of an SLA and the corresponding relationships

The conceptual model is composed of the following entities:

1. SLA: The SLA includes basic data, such as the title of the SLA, the corresponding ID, the type of application (i.e., smart home, smart health, etc.) and the start and end dates.

2. Party: This part describes an individual or group involved in the SLA and it usually includes the name of a service provider and/or consumer or a third party [166]. For example, in an RHMS, the parties could be the hospital management group, patients, the network provider and the Cloud resource providers.

3. SLO: The SLO provides the quantitative means to define the level of a service that a consumer can expect from a provider. The SLO quantifies the required value of a QoS metric. It expresses the objective(s) of an agreement for both the application and any involved services and infrastructure resources. For example, an SLO (at the application level) of an RHMS could be the response to urgent cases within `Y` time units. The QoS metric in this example is `response time`, and the constraint is less than `Y` time units. Furthermore, SLO parameters can be used to specify an SLO for low-level services. For example, for a data-ingestion service, an SLO can be: *ingest data with latency less than Z time units*. For an infrastructure resource such as the CPU of a VM, an SLO can be: *CPU utilisation is greater than 80%* .

4. Workflow Activity: IoT applications have certain activities that must be considered as part of the application requirements to function correctly. For example, in an RHMS, one of the possible workflow activities include capturing interesting data, analysing real-time data, and storing interesting results in a database (e.g., SQL or NoSQL). In general, workflow activities mostly include:

    - Capturing events of interest
    - Examining the captured events of interest on the fly
    - Filtering the captured events of interest
    - Aggregating the captured events of interest
    - Ingesting data from one or more data resources

- Small-scale real-time data analysis
- Large-scale real-time data analysis
- Large-scale historical data analysis
- Storing structured data
- Storing unstructured data

5. Service: This concept covers the main services that can be run/deployed to perform a certain functionality. To achieve SLOs at the application level, it is important to establish adequate cooperation between particular services under the SLO constraints. For example, in an RHMS, to detect urgent cases within Y time units, it is necessary to transfer data from sensors to the ingestion service using networking service and to process data on the fly using a stream-processing service while respecting the time limit.

   Here, we list the most common services that can cooperate to deliver an IoT application.

   (a) Sensing service: This service collects data using IoT devices and it sends the collected data through a communication protocol to a higher layer. The sensing service specifies the type of data and when to collect the data. For example, in an RHMS, a heartbeat sensor attached to the chest and an accelerometer as a hand-wrist device reflect a patient's health state continuously or periodically, based on what has been specified within the SLA for the service.

   (b) Networking service: This service transfers the collected data from one layer to another. For example, in an RHMS, a home gateway uses the network to deliver collected data to the next layer for further analysis under certain bandwidth requirements.

   (c) Ingestion service: This service ingests data from many data producers and then forwards the data to subscribed/interested destinations such as storage and analysis services under certain requirements, such as throughput limit.

   (d) Batch-processing service: A batch-processing service receives data from resources such as ingestion layers, appends them to a master data

set and then provides batch views. For example, in an RHMS, to identify urgent cases, it is important to run machine-learning algorithms on historical patient records to recognise patterns regarding certain health issues and to establish a predictive model. The predictive model can be used later with the real-time data of current patients to detect particular health issues. Batch views can be computed/queried within response time constraints, as specified by consumers/subscribers.

(e) Stream-processing service: This service processes incoming data from data resources such as an ingestion service to complete real-time tasks. For example, collected data are processed on the fly, and if the analysis shows an abnormality such as a high heart rate, then appropriate action is required, such as sending an ambulance. However, to exploit real-time data to the greatest extent possible, consumers/subscribers can specify certain requirements such as the maximum acceptable latency for computing/querying real-time views.

(f) Machine learning service: This is a service that applies different machine-learning algorithms for different purposes, such as providing predictions and extracting different dimensions of knowledge from collected data. For example, the service may apply a machine-learning algorithm to historical data collected from previous heart attach incidents as training data to create a model. The model can play a part in predicting new heart-attach incidents based on incoming real-time data. This approach may prevent disasters from happening or at least reduce damage by warning people in advance.

(g) Database service (SQL and NoSQL databases): This service is used by other services such as ingestion, batch and stream-processing services . It is used to store or retrieve data for batch views and/or real-time views as intermediate or final data sets. Consumers can provide their requirements, such as setting a query response time, and specify whether data encryption is required.

6. Infrastructure resource: This concept covers the required hardware for computations, storage and networking, which are essential for deploying/running the above-mentioned services. The infrastructure resources

can be IoT devices, Edge resources and/or Cloud resources.

(a) IoT devices: These devices has the ability to sense to reflect the physical world, then actuate and execute actions in some cases.

(b) Edge resources: These resources allow data processing to take place at the Edge of the network and they include various types of resources, such as border routers, set-top boxes, bridges, base stations, wireless access points and Edge servers [305]. These examples of Edge resources, with specialised capabilities [305], can be used to support Edge computations.

(c) Cloud resources: These resources provide infrastructure as a service (IaaS) and are mostly located geographically far from the source [305].

The relationships between the above entities, depicted in the conceptual model (Figure 3.7), are as follows. There is a one-to-many relationship between the `SLO` and the `SLA` entities to express the SLO constraints at the application level. Therefore, each `SLA` entity has a composite relationship with `SLO` entity. An example of an `SLO` at the application level could be the *end-to-end response time of an application should be less than Y time units*. Furthermore, `SLA` has a composite relationship with `Party`, since parties can play part in providing a service, consuming a service and/or playing third-party roles (e.g., to monitor a service).

Additionally, an IoT application has a set of workflow activities (e.g., capture an event of interest (EoI) or analyse real-time data) that cooperate to deliver the application. Therefore, there is a composite relationship between the `SLA` and `WorkflowActivity` entities. Each workflow activity requires a service (e.g., a sensing service, networking service, or stream-processing service). Each service is deployed on one of the infrastructure resources (for example, an IoT device, an Edge resource, or a Cloud resource). Furthermore, each one of the services (e.g., sensing is a service) and infrastructure resources (e.g., VM is an infrastructure resource) can have an SLO/SLOs. For example, maximising the level of data freshness could be an SLO for sensing services, and maximising CPU utilisation could be an SLO for a VM. Furthermore, each one of the services and infrastructure

resources can have zero or more configuration parameters (e.g., the sample rate of the sensing service and number of CPUs per VM of an infrastructure resource). Therefore, there is an association relationship between `InfrastructurResource` and `Service` and a composite relationship between the `InfrastructurResource`, `Service`, `SLO` and `ConfigurationRequirement` entities. The dashed rectangle in the conceptual model (See Figure 3.7) has a set of predefined data types which are defined as enumeration.

Figure 3.8 associates the concepts presented in the conceptual model with examples to illustrate the relationships between infrastructure resources, services, configuration requirements and SLO concepts. For example, "capture event of interest" is a possible workflow activity in an RHMS and it requires a sensing service. The sensing service has SLO constraints such as the required level of data freshness. The sensing service will be deployed/hosted on an IoT device. Therefore, it is important to consider the requirements of the IoT device, such as its type (e.g., sensor or RFID), the mobility of the device (e.g., fixed or mobile), the communication mechanism (e.g., pushing data or pulling data) and the battery life. The same conditions are applied for the "filter a captured event of interest" activity, which is performed at the Edge of a network to filter data and utilise network bandwidth by neglecting uninteresting data. This task uses certain devices, such as a mobile phone or raspberry pi. Each of these devices has specific computational capabilities, such as a given CPU speed and memory size. Furthermore, to perform the "real-time data analysis" activity, a stream-processing service can be used with certain requirement constraints, such as low latency and certain configuration requirements, including the specification of the window type as a time-based window or event-based window. The stream-processing service can be deployed on a Cloud. Thus, certain requirements related to a Cloud resource can be specified, such as the number of VMs and the acceptable percentage of CPU utilisation. Due to the important of specifying requirements of each involved service and the infrastructure which deploys that service and unifying used vocabularies, the next section identifies the related vocabulary terms that can be used for specifying QoS and configuration parameters of common services and infrastructure resources.

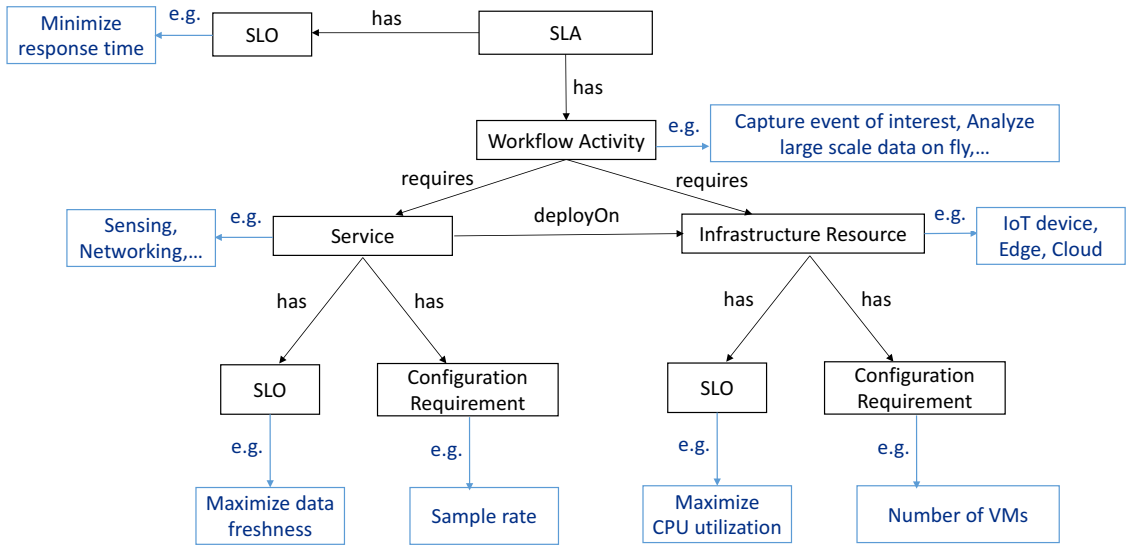

Fig. 3.8 Conceptual model with examples to illustrate the relationships between the key concepts of an SLA for the IoT

3.4 Vocabulary Terms of the Configuration Parameters and QoS Metrics

In this section, we cover in depth the "service" and "infrastructure resource" concepts with their sub-classes depicted in Figure 3.7. We describe the "service" and "infrastructure resource" concepts below with some of the related QoS metrics and configuration parameters.

We search the literature to collect vocabulary terms that are related to the QoS metrics and configuration parameters. The reason behind considering the terms related to configuration parameters is the strong correlation between the QoS and the configuration parameters. For example, the data publishing rate, as a configuration parameter, affects the data freshness as a QoS metric. This step comes after specifying the main components of the IoT reference architecture; then, the vocabulary terms that can be used to express consumer requirements are identified for each component. We believe that identifying domain-specific terms is the first step in providing unified/standardised vocabularies to mitigate

the risk that can be caused by the ambiguity between the different providers who cooperate to deliver an IoT application (Section 3.4.2 and Section 3.4.1).

3.4.1 Infrastructure Resources

Infrastructure resources include the type of the infrastructure resource used to deploy/host a service. An infrastructure resource can be an IoT device, an Edge resource or a Cloud resource. In the following, we give more details about the type of infrastructure resource and their related vocabulary terms:

IoT devices

IoT devices consist of heterogeneous sets of devices such as sensors that capture information about the physical world by sensing some physical parameters of interest or detecting other smart objects [137]. There are several QoS metrics related to perception layers, such as the optimum number of active sensors, sensor quality, energy consumption, data volume, trustworthiness, coverage and mobility [77, 214, 280].

Although some of these identified metrics may be inconsiderable for a single IoT device [280], they are not trivial when considering the number of deployed devices that cooperate to deliver a service. For example, a sensor with a power consumption value equal to 0.9 watt-seconds seems fine, but when a network of hundreds of sensors is deployed, the cumulative value of the power consumption makes a difference [280].

IoT communication protocols can be varied in their communication range, bandwidth and power consumption. Thus, it is important to consider support for different types of protocols, and the most appropriate type that satisfies the application requirements should be selected. For example, if the power consumption is the most important key requirement, then ZigBee, as a communication protocol that can be characterised as a low power consumption protocol [131], should be used. Alternatively, WiMax is a protocol that provides a high communication bandwidth. Therefore, it is essential to select devices that support the preferred communication protocol. Some of the available communication protocols are Bluetooth, WiFi, ZigBee, 6LowPAN, Cellular, ANT, Z-Wave, Thread, WiMax and

NFC [1]. Table 3.1 lists a number of vocabulary terms that can be used to express the requirements related to IoT devices.

Table 3.1 Terminology/vocabulary definitions related to IoT devices

Terminology	**Definition/Description /Example**
Device accuracy	Description of how well the device reflects an interesting event correctly.
Device precision	Description of how precisely the device reads an interesting event in a stable manner.
Type of device	For example, sensors and RFID tags.
Number of devices	The number of deployed devices.
Mobility of devices	Specification of whether the device is fixed or mobile (this feature affects network coverage).
Communication mechanism	The mechanism of pushing/pulling data to/from the next layer. This mechanism can be a built-in hardware feature, a software feature or both.
Communication technology	The communication protocol with other devices that are supported, such as by WiFi and Bluetooth. This technology can be a built-in hardware feature, a software feature or both.
Battery life	Battery life is a measure of battery performance and longevity, which can be quantified in several ways: as the run time on a full charge, as the milliampere hours estimated by a manufacturer, or as the number of charge cycles until the end of useful life.
Warranty period	The time period in which a purchased device may be returned or exchanged.
Storage size	The storage size of an IoT device that can be used to store data.
Memory capacity	The maximum or minimum amount of memory an IoT device has.
CPU capacity	The capability and speed of a processor, which reflect how many operations it can perform within a given amount of time.

Edge resources

In the Edge layer, intelligent computation abilities are allocated to Edge resources (a gateway, server, etc.) to improve performance and reduce unnecessary

[1] See [148] for further details and a comparison of communication protocols

data transfers to Cloud data centres. Sensitive and personal data, data management and control tasks are moved to the Edge to be managed in a secure and private manner [170]. Edge resources mostly include border routers, set-top boxes, bridges, base stations, wireless access points, Edge servers, etc. These examples of Edge resources can be equipped to support Edge computations with specialised capabilities [305]. Certain type of IoT devices (e.g., sensors) requires a gateway to link IoT devices with the Cloud layer, in this case, sensors have lightweight functionality [96]. Other type of IoT devices (e.g., smart sensors) can work without a gateway if the they have the ability to communicate directly with the Internet. Furthermore, for a more cost-effective approach with typical sensors that do not have gateway capability, it is possible to use many-to-one mapping. Many-to-one mapping is mapping many sensors to one gateway to allow data transferring by adding TCP/IP connectivity. [96].

Smart gateways can handle resource constraints on the processing power, power consumption and bandwidth of connected devices by allowing constrained devices to outsource some functionalities to the gateway. These gateways can be provided with local databases for temporarily storing sensed data, as well as enhancing data fusion, aggregation and internal device communication [396]. When specifying the QoS for an application, it is necessary to decide whether to deploy typical sensors and a gateway or a smart sensor. For example, using smart sensors (a smart sensor (with some processing capabilities) can behave as an IoT or an Edge resource) reduces the delay that is required for transferring data to the Cloud layer, which might be located at a distant position, and the data can be processed within the Edge resources instead of forwarding them to the next layer.

Some configuration parameters can affect the overall QoS of an IoT application. For example, the data publishing rate at the gateway is a concern because an increase in this rate might cause the the ingestion service to be "overloaded", which then causes messages to be dropped [61]. Another configuration parameter is the buffer/storage size, which plays a significant role in the performance of an IoT gateway. For instance, [60] proposes a multi-threaded gateway and considers different values for different parameters, including different buffer sizes to enhance gateway performance when evaluating the proposed model.

Table 3.2 Terminology/Vocabulary definitions related to Edge infrastructure resources

Terminology	**Definition/Description /Example**
Availability	The ratio of the time that the resource is functioning as expected and ready for use divided by the total run time.
Type of device	For example, a mobile, raspberry pi, or server devices.
Gateway throughput	The amount of data transferred through the gateway per second.
Gateway delay	The delay in data collection from nodes.
Publishing rate	Specifies when data need to be sent.
Number of devices	Total number of devices within the Edge infrastructure.
Mobility of devices	Specification of whether a device is fixed or mobile (this feature affects network coverage).
Communication mechanism	The mechanism of pushing data to the next layer or pulling data from the next layer; it can be a built-in hardware feature, a software feature or both.
Communication technology	The communication protocols with other devices, such as the communication protocols based on WiFi and Bluetooth. Such protocols can be a built-in hardware feature, a software feature or both.
Storage/buffer size	The buffer/storage size that can be used to buffer/store data due to limited throughput for out-coming data or to buffer/store data until delivery confirmation is received.
Memory capacity	The maximum or minimum amount of memory an Edge resource is capable of having.
CPU capacity	The capability and speed of a processor which reflects how many operations it can perform within a given amount of time.

Table 3.2 lists a number of vocabulary terms that can be used to express the requirements related to Edge resources.

Cloud resources

Most Cloud data centres are distributed internally across several physical data centres. As a result, many Cloud providers not only provide fault tolerance for a single machine or single rack but also provide resilience for full data centre failures, which yields a high level of reliability. Cloud providers supply computer resources on an on-demand basis. This approach quickly enables (typically in minutes) an arbitrarily large number of computing nodes to be accessed with

scale-up and scale-down possibilities [537].

Cloud resources can have one or more than one SLO; for example, an SLO can be "CPU utilisation should be more than 80%". Furthermore, a Cloud resource can have a configuration parameter, such as a number of vCPUs. Most Cloud systems provide a variety of storage features, such as those for the storage bandwidth, size, cost, latency, and access control for different storage types, including local instance storage, distributed block storage, distributed file systems and object (Binary Large OBjects e.g., (BLOB) storage. These various services can lead to very different choices regarding software design, depending on the system or application requirements [537]. Table 3.3 lists a number of QoS and configuration parameters for Cloud resources. There are different types of instances, e.g., instances with more RAM versus more storage, or with specific hardware components, such as GPUs or FPGAs [537].

Table 3.3 Terminology/Vocabulary definitions related to Cloud infrastructure resources

Terminology	**Definition/Description /Example**
Availability	The ratio of the time that the resource is functioning as expected and ready for use divided by the total run time.
CPU utilisation	Percentage representing how the CPU is being utilised.
Outage length	The length that the resource is not available.
Throughput	The data transfer rate to and from a Cloud resource per second.
Storage size	Available disc space for data storage purposes.
Storage bandwidth	Measure of the capacity to transfer data between a service and storage.
Storage type	Type of storage for a service (e.g., local SSD or local HDD).
Input/output storage operations	The specified number of input/output operations for storage.
Access protocols	Cloud access protocols such as SSH and SSL.
Memory capacity	The memory capacity is the maximum or minimum amount of memory a computer or hardware device is capable of having or the amount of memory required for a program to run.
Network bandwidth	Network speed among the internal service nodes involved (e.g., 100BASE-T, 100BASE-SX).
vCPU capacity	The capacity of each virtual central processing unit (vCPU) which reflects how many operations a vCPU can perform within a given amount of time.
No. of vCPUs	The number of vCPUs per VM.
No. of cores per VM	The number of cores per VM.
Vertical scale-down limit	The minimum number of CPUs if scaling is not automatic.
Vertical scale-up limit	The maximum number of CPUs if scaling is not automatic
Horizontal scale-up limit	The maximum number of VMs if scaling is not automatic.
Horizontal scale-down limit	The minimum number of VMs if scaling is not automatic.
Replication factor	The number of copies of data that one wants the cluster to maintain.

3.4.2 Service Concept

To achieve SLA constraints at the application level, it is important to ensure an adequate cooperation between some services under the SLO constraints. Therefore, we use the service concept to capture the name of the required services. A service has one or more SLO constraints and configuration requirements. There is a number of possible services which includes, but not limited to, sensing, networking, stream processing, batch processing, database, and machine-learning algorithm services. Each one of the previously mentioned services can have one SLO or more; for example, an SLO for stream-processing service can be "minimising latency to be less than 5 time units". Furthermore, each of the previously mentioned services can have configuration requirements; therefore, there is a relationship between the service and configuration requirement concepts as depicted in Figure 3.7. For example, a service such as stream processing can specify a requirement related to the “window size” (the window size is a configuration parameter). In the following section, we list the most common services that can cooperate to deliver an IoT application.

Sensing Services

A sensing service is responsible for collecting data from IoT devices and sending the collected data through a communication protocol to another layer. The sensing service specifies the number of sensors, type of sensors, and sampling rate. In an RHMS, for example, in order to provide a sensing service, we need to specify the type of sensors associated with a patient, such as a heartbeat sensor attached to the chest and an accelerometer on the hand/wrist to reflect the patient’s activities. A sensing service is associated with different parameters that play a significant role in the overall QoS of an IoT application.

For example, different applications require varying sampling rates depending on their criticality. The sampling rate determines the frequency at which an observed phenomenon is measured by a sensor (e.g., 5 Hz) [214]. Moreover, gaps in historical data can cause IoT applications to behave unexpectedly, which affects the final outcome and can lead to a bad user experience. Therefore, the IoT platform must attempt to maximise data freshness [481]. The importance of the freshness parameter from the perspectives of both producers and consumers

has been recently discussed in [391]. The authors argued that for transient IoT's content, both data of interest and data packets should have a certain freshness to perform accurate caching and retrieval operations. Additionally, old content is automatically discarded from data storage as a consequence of the freshness requirement [188]. Moreover, data freshness is one of the security requirements in the IoT because if an attacker first captured data and resent them, the data would become old [385][176].

Another metric is data quality, which is a complicated metric since it relies on other metrics, such as data accuracy [315]. Data accuracy, itself, is affected by data freshness and precision [244], reflecting the high interdependence among metrics. Furthermore, application objectives such as reducing energy consumption and non-functional properties are interdependent. For example, increasing the sampling rate plays a significant role in enhancing data freshness, which in turn improves the information quality; however, this change decreases battery life (i.e., increases energy consumption). Table 3.4 lists some of the vocabulary terms that can be used to express the QoS constraints and configuration parameters relevant to sensing services.

Table 3.4 Terminology/Vocabulary definitions related to sensing services

Terminology	**Definition/Description /Example**
Availability	The ratio of the time that the service is functioning as expected divided by the total run time.
Data freshness	The age of sensor data because data cannot always be transmitted in real time/near-real time
Sampling rate	The rate at which a sensor measures an observed phenomenon (e.g., 5 Hz). Different applications require different sampling rates based on their criticality.
Data accuracy	The error rate of data. It is possible to specify the average number of errors over a given time period.
Data integrity	Data integrity reflects the degree to which data have been maintained or altered.
Data type	e.g., Capturing weather temperature or humidity.

Networking Service

Networking service is used for passing the collected data from one layer to another. They also provide a bidirectional connection for cases in which an instruction needs to be sent to one or more devices. For example, in an RHMS, gateways use the network to deliver collected data to the Cloud for further analysis. A networking service is also used when a command is sent back to a sensor, for example, to reconfigure the sampling rate, to collect more data or check a patient's status. Thus, a network service is responsible for transferring data between an IoT and an Edge resource [97]. Furthermore, in some cases, an IoT device has the ability to communicate without needing a gateway; in such a case, the networking service is used to immediately connect the device to Cloud services (e.g., ingestion service and/or stream-processing service). The quality of the network is crucial to being able to deliver the data within the acceptable time limit before they lose value. Therefore, it is critical to consider the QoS requirements of the network layer.

QoSs have been extensively researched in the field of network communications and have well-defined and measurable characteristics, such as throughput, jitter or packet loss [244], which impact the network delay [264] [121] [237] [76] [141]. Table 3.5 lists some of the vocabulary terms that can be used to express QoS constraints and configuration parameters that are relevant to networking services.

Table 3.5 Terminology/Vocabulary definitions related to networking services

Terminology	Definition/description/Example
Availability	The ratio of the time that the network is fully operational as expected and ready for use, divided by the period of time.
Link bandwidth	The maximum amount of data that can be transferred through a link per second.
Network delay	The delay in data transmission.
Data-in rate	The amount of incoming data per time unit.
Data-out rate	The amount of outgoing data per time unit.
Jitter	The time delay variance between data packets over a network in milliseconds (ms).
Packet loss rate	The ratio of the number of packets lost to the total number of packets sent. Each packet has a deadline for execution, and if meeting this deadline is not possible, the scheduler tries to minimise the number of packets lost due to deadline issues.
Data integrity	Data integrity reflects the degree to which data have been maintained or altered.

Ingestion Services

An ingestion service allows data to be ingested from many data producers [399] and then forwarded to subscribed/interested destinations, such as a storage service, analysis service and/or application.

An ingestion service can be associated with different parameters, such as configuration requirements (e.g., the number of servers/nodes and compression/decompression support) and SLO constraints (e.g., maximising throughput and minimising latency).

In an ingestion service, data often come from a variety of sources, including web logs, databases, various kinds of applications, etc., making it difficult to understand what sort of data the system will ingest. One alternative is to use big data (BD) software, which can collect and aggregate data from various sources. Projects such as Flume [2] and Scribe [3] enable the collection, aggregation and transfer of large quantities of log information from many distinct sources to a centralised data storage centre [484].

[2] http://flume.apache.org/
[3] https://github.com/facebookarchive/scribe/wiki

Data retention is one of the parameters that service consumers need to specify to indicate how long data can be stored before they are deleted. Therefore, the data rate and data retention time are interdependent since they represent key factors related to resource storage. For example, in Kafka [4], the data rate of a partition is the rate at which it generates information; in other words, it is the average size of the message multiplied by the amount of messages per second. The data rate indicates how much retention space is needed in bytes for a given amount of time, in order to ensure retention. If there is a lack in knowledge regarding the data rate, the retention space needed to meet a time-based retention goal cannot be calculated properly [314].

Messaging systems provide some replication-related functionality to improve various factors, including reliability, fault tolerance and accessibility for replicating data/messages on different servers. For example, replication is used by default in Kafka; even unreplicated topics are implemented as replicated topics [218]. Data encryption, data compression and delivery guarantee mechanisms are application dependent. Thus, for example, if providing a low-latency solution is important, then providing data encryption and delivery guarantee mechanisms may cause delays. Furthermore, if reliability is important, then providing a delivery guarantee mechanism that ensures that messages/data/requests are delivered using the ingestion service is crucial. In other cases, when throughput is highly prioritised over latency, data compression is a key concern. The available messaging systems provide compression, encryption and delivery guarantee mechanisms. As an example, Amazon Kinesis Data Firehose [5] enables the compression of information before it is delivered, and it supports the GZIP, ZIP and SNAPPY compression formats [53]. Amazon Kinesis Data Firehose, also, allows for data encryption using the AWS Key Management Service [53]. RabbitMQ [6] and Kafka both offer durable messaging guarantees. Both offer at-most-once and at-least-once guarantees, but in very restricted situations, Kafka provides precisely once guarantees [480]. Table 3.6 lists some of the vocabulary terms that can be used to express some of the QoS constraints and configuration parameters that are relevant to ingestion services.

[4] https://kafka.apache.org/
[5] https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html
[6] https://www.rabbitmq.com/

Table 3.6 Terminology/vocabulary definitions related to ingestion services

Terminology	Definition/Description /Example
Availability	The ratio of the time that the ingestion service is functioning as expected divided by the period of time.
Throughput	The amount of data transferred through the messaging platform per second.
Latency	The time required to process a single input/output transaction before forwarding it to its destination within the ingestion service framework.
Data-in rate	The amount of incoming data per time unit.
Data-out rate	The amount of data output per time unit.
Data retention time limit	The limit of how long data can be saved in the ingestion layer.
Publishing rate	Rate at which data is sent from a message broker per time unit.
Storage size	The amount of storage that can be used to store data due to limited throughput constraints, considering the amount of incoming data, to store data until delivery confirmation, or to store data during the specified retention time.
Replication factor	How many replicas can be stored.
Data compression support	A Boolean value that expresses whether data can be compressed/ decompressed depending on the requirements.
Data encryption support	A Boolean value that expresses whether data can be encrypted/decrypted depending on the requirements.
Delivery guarantee mechanism	It reflects if data have been delivered to the destination. The network bandwidth is affected if the type of delivery guarantee mechanism requires sending an acknowledgement back to the data producer.
Data integrity	Data integrity reflects the degree to which data have been maintained or altered.
Name of ingestion framework	e.g., RabbitMQ, Amazon Kinesis Data Firehose, Flume, Scribe

Stream-processing services

A stream-processing service refers to processing incoming data from different data sources and/or ingestion services to compute real-time views. Furthermore, real-time views can be combined with saved computed batch views using a database framework (such as Cassandra [7]) to answer some questions that rely on both real-time views and batch views. In an RHMS, data can be collected using different sensors, such as wearable accelerometers, that can be augmented by distributed-motion sensors for activity recognition purposes [379]. If the collected data show abnormality for a given activity, such as an elderly person falling down, then an appropriate action, such as sending an ambulance, is required. However, applications such as RHMSs rely on real-time data, therefore any delay in data processing could cause the data to lose their value.

High throughput and low latency are very important QoS requirements in stream processing. If incoming data are not analysed in real or near-real time, then the action taken may not be appropriate since actions are based on data that are no longer considered real-time/near-real-time data due to the delay. Another important metric is data completeness, which "measures the percentage of incoming stream data that are used to compute the query results" [515]. To illustrate the concept of data completeness, consider a data stream with a number of incoming tuples. In the ideal case, the query should be performed using a large sliding window, e.g., containing 30 tuples; however, due to resource constraints, 15 tuples are sampled and used to execute the query, representing 50% of the 30-tuple window size. The sampling method decreases the query data completeness to 50% [515]. Furthermore, another QoS metric is the miss ratio, which "evaluates the number of queries that are not completed within the given time constraints" [515].

In addition, single-point resource estimation is insufficient to handle stream processing workloads in which information flows endlessly through the operator graph and yields changes in performance and resource demands. Therefore, to illustrate the effects of certain configuration parameters on performance and resource usage, consider the work in [377] as an example. Patel et al [377]

[7] http://cassandra.apache.org/

present a novel method using mixed density networks, a mixed structure of neural networks and mixed models to estimate the resource usage of data stream processing workloads in the Cloud. To train the proposed model, a set of features is used as the model input; the set includes the size of windows that can be expressed in time units (second) or tuple units (number), the sliding value of the window type, the average arrival rate of tuples (tuple/second) to query, the total number of nested sub-queries and the operator type. The set of features is customised based on the prediction goal because the impact varies with respect to the CPU and memory. A feature that is correlated with memory consumption may not be correlated with CPU usage. For example, the selection results for features suggest that the size of the window has an insignificant effect on the prediction of CPU use but a notable influence on the prediction of memory use [377].

Furthermore, the QoS requirements of stream processing are affected by other configuration parameters, such as the window size and query size; in addition, the choice of a stream processing framework affects the QoS. For instance, selecting a framework (such as Spark streaming) [8] that stores data before processing affects the latency level; Apache storm [9] can process data immediately with no need to store them first to save time and reduce latency [135]. Table 3.7 lists the key terms/definitions related to stream-processing services to express the requirements for both QoS metrics and configuration parameters.

[8] https://spark.apache.org/streaming/
[9] https://storm.apache.org/

Table 3.7 Terminology/vocabulary definitions related to stream-processing services

Terminology	**Definition/Description /Example**
Throughput	The stream size processed per second.
Latency	The time required to process a single input/output transaction for a stream-processing service.
Data completeness	Measurement of "the percentage of incoming stream data that are used to compute the query results." [515]
Miss ratio	"Miss ratios measure the percentage of queries that are not finished within the given deadlines." [515]
Time-based window size	The size of the window with respect to the time required to process data that occur within the window.
Event-based window size	The size of the window based on a number of events/records/messages within a given window.
Sliding window	Determines the length of the window and the portion of the range that is retrieved when the window moves forward; the intervals can overlap. This value can be time based, count based or based on a hybrid scheme.
Tumbling window	A series of fixed-sized, non-overlapping and contiguous time intervals.
Micro batch size	Specification of the size of data that need to be buffered first before being processed; however, in stream processing, data are not required to be stored first. It is better if data are processed in active mode, which means that data are processed as they arrive and not when they are pulled.
Data arrival rate	Specification of how many data are expected to be received per second.
Write capacity	Specification of the capacity of writing in one go.
Read capacity	Specification of the capacity of reading in one go.
Replication factor	Expression of how many replicates can be stored.
Total number of queries	Specification of how many queries should be considered.
Data Compression support	A Boolean value that expresses whether data can be compressed/ decompressed depending on the requirements.
Data Encryption Support	A Boolean value that expresses whether data can be encrypted/decrypted depending on the requirements.
Data Integrity	Data integrity reflects the degree to which data have been maintained or altered.
Name of stream processing framework	e.g., Spark streaming, Apache storm

Batch Processing Services

A batch-processing service refers to receiving data from ingestion layers and/or other data sources, appending the data to the master dataset and then obtaining batch views; moreover, the computed batch views can be stored for inquiry purposes. Batch processing can be based on incremental algorithms or recomputation algorithms [316], considering the type of job that needs to be accomplished. For example, in an RHMS, if hospital management is interested in recording some statistics regarding detected urgent cases, one interesting statistic might be the total number of urgent cases that have been detected. The count function can then be applied using an incremental algorithm or recomputation algorithm. However, since the number of newly detected cases can be added to the previous calculated total number of detected cases, an incremental algorithm could be more suitable. The reason for choosing an incremental algorithm in this case is that the total number can be calculated without considering the entire dataset; this process avoids the need for additional computational resources since it only requires an increment step.

However, if the query must consider the whole dataset, for example a query regarding the average age of people who have a certain health issue, then whenever new cases arrive, there is a need to recompute the average considering all of the recorded ages, which requires a recomputation algorithm. Selecting the appropriate algorithm is important. Recomputation algorithms require computational efforts/resources to handle the master dataset, while fewer computational resources are required for incremental algorithms. However, a recomputation algorithm is more robust since it is human-fault tolerant because batch views are continuously recomputed [316].

In batch-processing services, the throughput and query response time are key QoS requirements in which users are interested. The related terminology/vocabulary definitions are used to express configuration requirements (such as the number of map and reduce tasks and the batch size). Furthermore, the choice of batch-processing framework affects the QoS. For instance, Hadoop [10] is a powerful batch-processing framework; however, it is not the appropriate choice

[10] https://hadoop.apache.org/

when there is a need to apply machine-learning algorithms because it requires data to be reloaded from the disk, which increases the latency; therefore, in this case, Apache Spark could be an ideal choice [135].

Furthermore, [203] presented a mathematical model for the optimum number of map tasks in MapReduce resource provisioning, to estimate the optimum number of mappers based on the resource specifications and data set size. The MapReduce library divides input data into several InputSplits[11]. A map task reads an InputSplit and processes the InputSplit using the user-defined map function. The map function takes input key/value pairs and creates a set of pairs for an intermediate key/value. The mapper memory buffers the intermediate key/value pairs. If the size of the data set reaches the memory buffer threshold, intermediate key/value pairs are stored on the local disc and partitioned to reduce the task requirements using the hash function. The reduce tasks involve reading and sorting steps for the intermediate data and group data with the same key. Then, the key and intermediate value sets are sent as inputs to the reducer to be written to the reducer's memory, and the reduce function is invoked [483]. The output of the reduce function is concatenated and then written to the output file [203]. The MapReduce model and Hadoop Open Source Implementation are effective for large data processing tasks. They are inherently built for batch processing jobs with high throughput requirements [425]. Throughput, as a QoS metric, indicates the number of MapReduce jobs completed per time unit (e.g., minutes) [147]. Furthermore, it should be noted that the number of map tasks can be used as a cost estimator, as applied in [147]. Table 3.8 lists the terminology/vocabulary definitions related to expressing the QoS metrics and configuration parameters of batch-processing services.

[11] InputSplit represents the data which can be processed by an individual Mapper

Table 3.8 Terminology/Vocabulary definitions related to batch-processing services

Terminology	**Definition/Description /Example**
Throughput	The number of batches that can be processed per second.
Response time	The time required to process a submitted job and receive a response.
Batch size	The limit on the size of each batch that is submitted to be processed.
No. of batch jobs	The number of submitted batch jobs.
Process running frequency	Specification of how frequently the process needs to be run, e.g., twice per hour.
Max. memory of the map task	Amount of memory assigned to the map task.
Max. memory of the reduce task	Amount of memory assigned to the reduce task.
No. of mappers	The number of mappers.
No. of reducers	The number of reducers.
Write capacity	The capacity of writing in one step.
Read capacity	The capacity of reading in one step.
Replication factor	Expression of how many replicas can be stored.
Total number of queries	Expression of how many queries should be considered.
Data compression support	A Boolean value that expresses whether data can be compressed/ decompressed depending on the requirements.
Data encryption support	A Boolean value that expresses whether data can be encrypted/decrypted depending on the requirements.
Data integrity	Data integrity reflects the degree to which data have been maintained or altered.
Name of batch processing framework	e.g., Hadoop

Machine Learning Services

A machine learning service refers to a service that permits the use of various machine-learning algorithms to predict the purposes and different dimensions of knowledge from the collected data. For instance, a machine algorithm can be applied to historical data collected from patients with heart attack incidents to obtain training data. Then, the training data can be used to create a model to predict heart attack cases based on incoming real-time data, which can prevent emergencies from happening or at least reduce patient damage by warning patients in advance.

In terms of practical needs, there are different QoS metrics, including speed, accuracy, etc., as in most topic detection and tracking (TDT) applications. Furthermore, different types of algorithms for machine learning affect accuracy and speed differently. The algorithm class reflects the type of algorithm, including classification, clustering, etc., whereas the algorithm name refers to the specific algorithm used, such as K-means, linear discriminant analysis (LDA) and naive Bayes [12]. Different algorithms, even if they are from the same class, can have different impacts on the performance of a system. For example, some clustering algorithms, such as the K-means and Canopy algorithms, differ substantially in their speed of execution; specifically, K-means has more than one iteration, while Canopy has only one iteration [488]. Table 3.9 shows a list of the main QoS metrics and configuration parameters that are related to machine learning services.

[12] Refer to [128] for further details about machine-learning algorithms

Table 3.9 Terminology/vocabulary definitions related to machine-learning algorithm services

Terminology	Definition/Description /Example
Accuracy	The accuracy of the analysis.
Class of ML	The name of the class in which an algorithm is classified. For example, supervised learning involves classification and regression algorithms, and the unsupervised learning class includes clustering and association algorithms.
Name of ML algorithm	Specifies the name of the algorithm required, such as logistic regression, decision forest, decision jungle, neural network, support vector machine, principal component analysis (PCA)-based anomaly detection, K-means, or naive Bayes.
Way to run the ML algorithm	Examples of this process are Sequential and MapReduce.
Data integrity	Data integrity reflects the degree to which data have been maintained or altered.

Database Services

A database service can be used for data retrieval with different services, such as ingestion, batch and streaming services. The database service stores incoming data as an intermediate or final dataset, a set of computed batch views or a set of computed real-time views. For instance, the incoming data can be initially stored, such as with Hadoop Distributed File System (HDFS)[13] in Hadoop, before any further processing. Then, the data can be retrieved for analysis or can be processed on the fly, and the derived results are stored in a database such as Cassandra.

Different types of databases are selected based on the purpose of the application and the required QoS. For example, in stream processing, data can be stored in databases that support low-latency read and write operations, whereas cases that require immutable data can use durable object storage platforms such as Amazon S3[14], which is preferable to other applications. Furthermore, to handle large amounts of data, a distributed database platform is employed, such as the available open-source distributed database Druid[15], which supports

[13]HDFS represents the storage component of Hadoop framework.
[14]https://aws.amazon.com/s3/
[15]http://druid.io/

data ingestion as well as queries with low latency, and Apache HBase[16], which supports the random and real-time reading/writing of large volumes of data. However, the selection of the appropriate platform is affected by several factors, such as the query response time [135]. Table 3.10 lists some of the most common QoS metrics and configuration parameters of database services.

Table 3.10 Terminology/vocabulary definitions related to database services

Terminology	**Definition/Description /Example**
Throughput	The queries that can be processed per second.
Response time	The time from when a user sends a request to when they receive a response.
Type of database	For example, SQL or NoSQL.
Type of NoSQL	For example, a key-value, document-based, graph-based, or column-based NoSQL.
Read error rate	The number of errors associated with reading attempts per time unit (seconds).
Cache hit ratio	The ratio of cache hits to misses, expressed as a percentage. A cache hit is when the data requested for processing are found in the cache memory. A cache miss is when the data requested for processing are not found in the cache memory.
Write error rate	Rate of errors associated with writing attempts per time unit (seconds).
Write capacity	The capacity of writing in one step.
Read capacity	The capacity of reading in one step
Replication factor	Expression of how many replicas can be stored.
Compression support	A Boolean value that expresses whether data can be compressed/decompressed depending on the requirements.
Data encryption support	A Boolean value that expresses whether data can be encrypted/decrypted depending on the requirements.
Data Integrity	Data integrity reflects the degree to which data have been maintained or altered.

[16] http://hbase.apache.org/

3.5 Evaluation

Our evaluation method is designed to introduce the participants to the conceptual model, discuss it and clarify any unclear points. Then, the participants can offer their opinions based on what has been introduced and their knowledge of the field, using a questionnaire.

Previous works, such as [476, 245, 227], did not mention any form of evaluation to their proposed conceptual model. However, in this section, we present our evaluation approach to assessing the proposed conceptual model. We have applied a goal-oriented questionnaire approach, and further details of the evaluation procedure and the results are presented in the following section.

3.5.1 Experiment

The main purpose of the conducted experiment was to evaluate the proposed conceptual model and to determine whether it meets the relevant predefined goals: generality, based on the coverage of general concepts that are common in IoT applications; coverability, or the extent to which IoT application requirements are covered, considering the main concepts that can be used within an SLA to express QoS constraints and configuration requirements.

3.5.2 Participants

The potential users of our proposed work are IoT administrators. Therefore, the research interests/topics of the participants in our experiment are mainly related to the IoT. The study was conducted with 14 participants; most of them are Ph.D. students who are working on topics related to the IoT, such as remote health and smart city applications. Their research interests included Cloud computing, Edge computing and networking. Table 3.11 provides a brief description of the research interests of each participant.

3.5.3 Procedure

The experiment was carried out following a well-defined procedure. Our evaluation method was a focus group followed by a questionnaire. The focus group approach has several advantages, for instance, the collection of in-depth in-

Table 3.11 A brief description of participants' research interests

Participant	**Research Interest**
1	IoT Workflow composition and configuration management
2	Topics related to the network layer of the IoT and the Cloud
3	Monitoring performance of BigData cluster in multi-Cloud
4	IoT data management and analytics
5	Remote health monitoring using the IoT
6	Research related to IoT projects
7	Real-time ambulance system
8	BigData workflow orchestration
9	Monitoring of building energy performance using the IoT
10	Security of the IoT
11	Fault tolerance in the IoT
12	Automating Computational Placement in IoT environments across heterogeneous platforms
13	Research related to IoT and Cloud projects
14	Blockchain and IoT

formation and the expansion and clarification of questions. Thus, we use the focus group approach to review the conceptual model and follow it up with a questionnaire to allow the participants to express their opinions.

To encourage the participants to take part in this evaluation, and to save their time, the questions were closed questions [17]. However, there was a comment textbox to allow the participants to comment and make suggestions, provide criticisms or give other feedback.

First, in the focus group[18], the participants received an introduction to the SLA and the reference architecture of the IoT, and a presentation was given on the conceptual model. The participants were allowed to discuss and comment on the conceptual model. A use-case was employed for scenario clarification purposes (RHMS). At the end of this period, the participants were asked to submit a written version of the completed questionnaire, in which there are three questions related to the conceptual model. Furthermore, there is a comment textbox to allow the participants to comment and make suggestions, provide

[17]Closed questions are questions that provide participants/respondents with pre-populated answer choices

[18]The size of the focus groups varied based on the availability of the participants

criticisms or give other feedback.

We present the participants with three questions that reflect the objectives that we aim to measure. There are positive and negative options associated with each question. The three questions related to the conceptual model are as follows:

- Overall, how satisfied or dissatisfied are you with our conceptual model?
- To what extent does the conceptual model cover your requirements?
- How satisfied are you with the conceptual model's generality?

3.5.4 Experimental results

We applied Likert scales, which are very common because they are one of the most popular ways of measuring attitudes, beliefs and behaviors. In contrast to binary questions, which only yield two answers, Likert-type questions provide more granular feedback.

This approach allows researchers to discover degrees of opinion that could make a real difference to understanding the feedback. Often, it can recognise places where developments could be made to the service or product. However, this type of questionnaire has weaknesses, such as acquiescence, meaning that the participants might consent to statements made in order to "please" the experimenter. However, to minimise this risk, the experimenter made it clear that providing names was optional. Furthermore, the experimenter provided negative options such as "Dissatisfied" and "Very dissatisfied" for each question in addition to other positive options such as " Satisfied". In addition, to minimize the influence of colleagues (such as avoid having cases where one participant is affected by his/her colleagues opinion when answering the questionnaire), each participant provided their feedback separately to prevent any external influence and on their own time to prevent time pressure.

Figures 3.9, 3.10 and 3.11 show the results based on the participants' answers with regard to the proposed conceptual model. Fifty percent of the participants described their overall satisfaction level as satisfactory, while the other fifty

percent were very satisfied. Regarding the generality of the conceptual model, more than 60% of the participants were very satisfied, and the rest were satisfied.

Regarding the coverability (capturing the main related concepts) of the conceptual model, more than 40% of the participants answered that the model provided full coverage, and the rest of the participants answered: “mostly covered”.

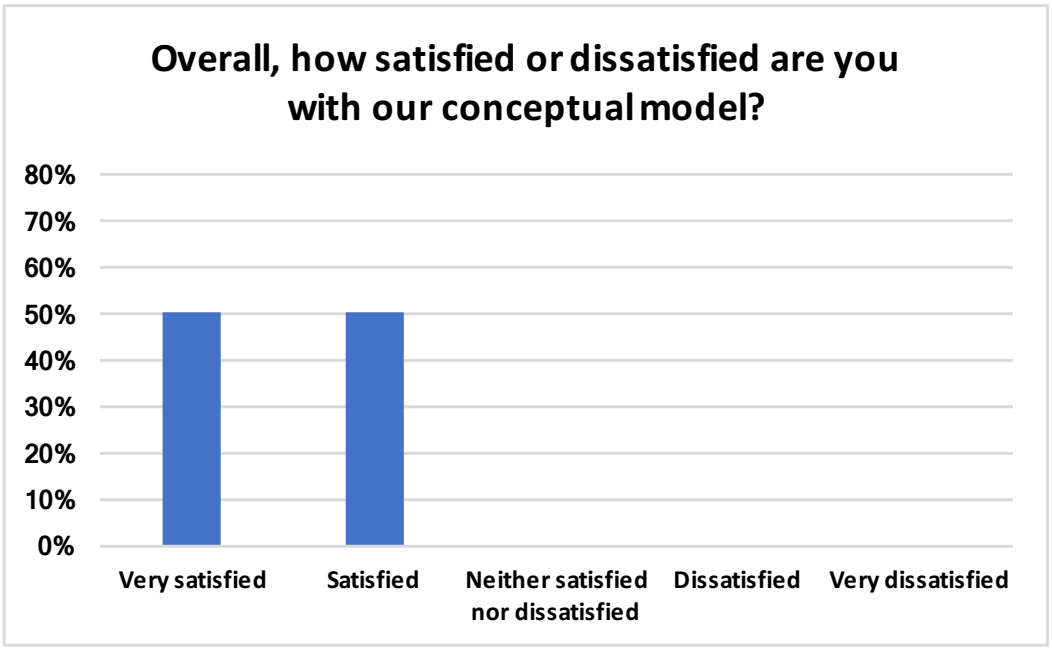

Fig. 3.9 Results of the evaluation: Satisfaction

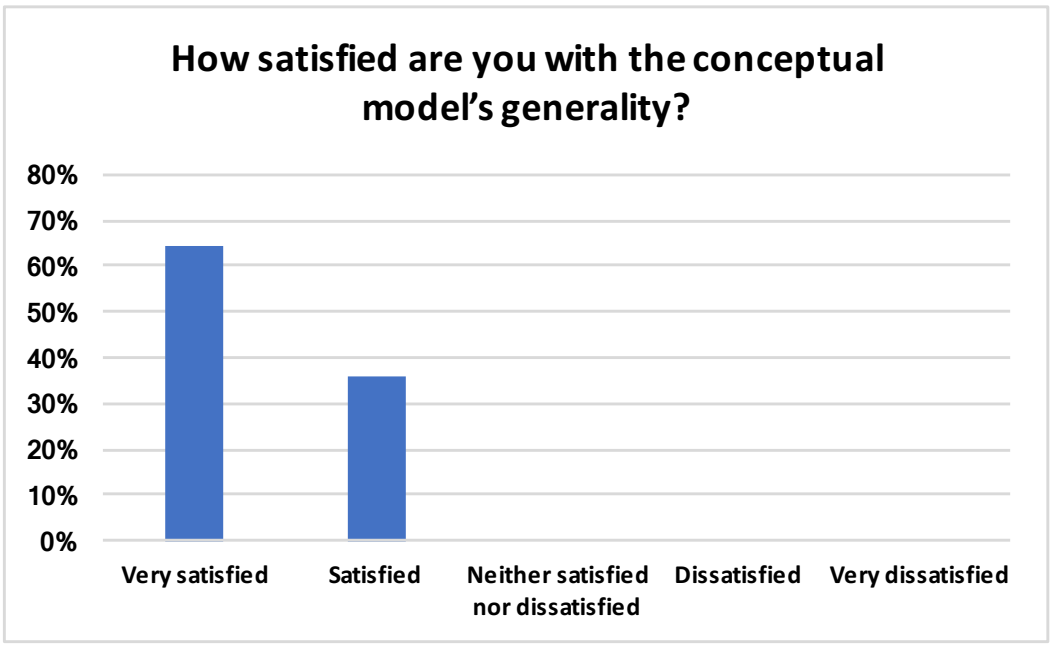

Fig. 3.10 Results of the evaluation: Generality

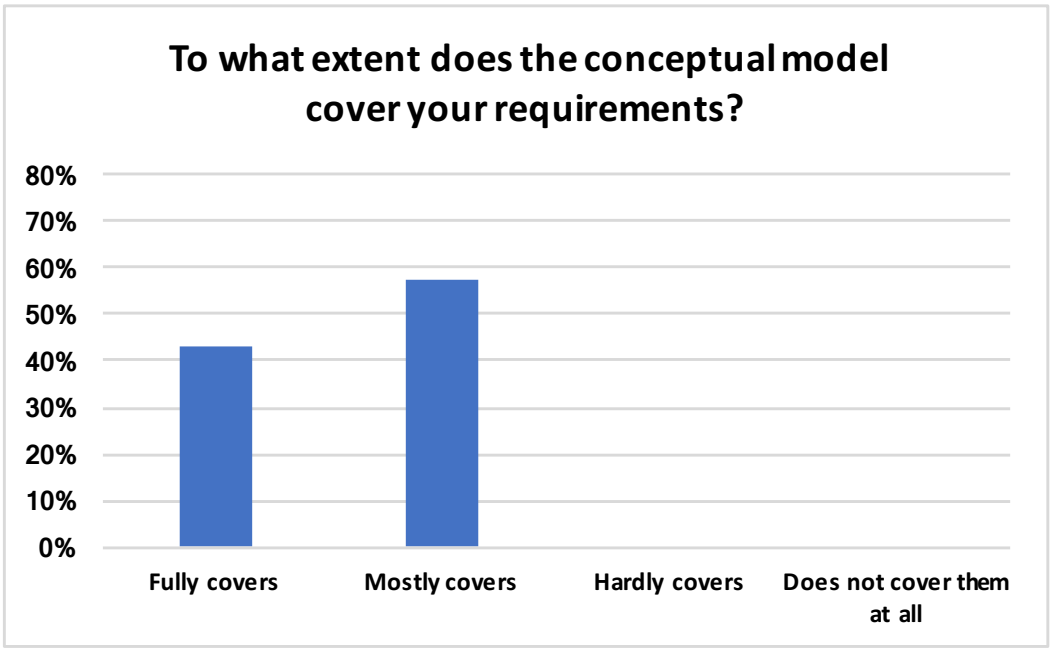

Fig. 3.11 Results of the evaluation: Coverability

3.5.5 Evaluation Analysis

As the answer for each question was based on a Likert scale, ordinal codes were assigned to the answers. For example, from very satisfied=1 to very dissatisfied=5, and from fully covers=1 to does not cover them at all=5. Percentages were used to explore the distribution of answers, while the median was computed to define the general tendency of the participants. The Wilcoxon test for one sample was used and it is a nonparametric test which we used due to the small sample size [132]. The Wilcoxon test was used, in this study, to examine whether there was a significant satisfaction with the conceptual model's generality (median<=2) and coverability (median<=2). P-value is used to decide if the difference is significant. A p-value of 0.05 was used as the threshold for significant results (0.05 is a rule of thumb, so the result is significant if p-value <0.05).

The results in 3.12 indicate that the participants were very satisfied with the conceptual model (median=1.50), at a very highly significant level (p-value<.001). Regarding the coverability,the participants believed that the conceptual model mostly covered the main concepts (median=2.00), at a very highly significant level (p-value<.001). The participants were satisfied with the generality of the conceptual model (median=2.00), at a very highly significant level (p-value<.001). There were a few comments regarding the concept names that describe resources, and it was suggested that "resources" be changed to "infrastructure resources". Furthermore, there was a comment regarding the presentation of the conceptual model as follows: "it would be better if it (the conceptual model) was represented in hierarchical view" and both comments are considered.

Table 3.12 Result of conceptual model using the Wilcoxon test

	Median	p-value (Wilcoxon test)	Decision
Overall, how satisfied or dissatisfied are you with our conceptual model?	1.5	<.001	Significant result
To what extent does the conceptual model cover your requirements?	2	<.001	Significant result
How satisfied are you with the conceptual model's generality?	1	<.001	Significant result

3.6 Conclusion and Future Work

In this chapter, we tried to overcome one of the end-to-end SLA specification challenges related to the heterogeneity of key QoS metrics across the computing environment. We proposed a conceptual model for an IoT-specific SLA. Then we identified domain-specific vocabulary terms that can be used as a starting point for an SLA specification, considering both the QoS constraints and configuration parameters across layers.

There is a limitation in the presented work related to the sample size of participants who evaluated the SLA conceptual model. However, the reason for the small sample size is that we sought participants with domain-specific knowledge, especially as we were looking to review our conceptual model with experts. In future work, we will try to extend the identified services and infrastructure resources and identify a list of vocabulary terms related to QoS metrics and possible configuration parameters. Furthermore, we will try to evaluate the proposed model with a larger sample size.

In the next chapter, we present an SLA grammar set derived from the work presented in this chapter.

Chapter 4

Service level Agreement Specification for IoT Applications

Overview

It is essential to consider the SLA specification as a first step towards SLA monitoring and management. We believe that current SLA specification formats are inadequate and unable to accommodate the unique characteristics of the IoT domain, such as its multi-layered nature. Therefore, this chapter proposes a grammar for the syntactical structure of an SLA specification for the IoT. The syntax is based on the proposed conceptual model (Chapter 3), which considers the main concepts employed for expressing the requirements of the services and infrastructure resources of an IoT application on an end-to-end basis.

To evaluate the proposed SLA specification, we conducted a user study with domain experts. The participants were researchers whose main research interests were related to Cloud computing, networking and the Internet of Things. The evaluation process was conducted by applying the goal/question/metric (GQM) approach to reflect user satisfaction with the identified vocabularies. The results show a high level of satisfaction with the generalizability and expressiveness of the considered vocabularies in terms of capturing the QoS metrics and configuration parameters of an IoT ecosystem.

4.1 Introduction

Specifying constraints within an SLA is essential because it protects consumer rights from any damage encountered during the contract period. Thus, it increases the level of trust between a service consumer and a service provider [432, 248]. Additionally, due to the multi-layer nature of the IoT and the huge dependency between cooperating layers [28], the individual SLA management mechanism for each layer of the IoT is inadequate. Therefore, this chapter proposes a grammar for the syntactical structure of an end-to-end SLA specification for the IoT. The syntax is based on the proposed conceptual model presented in Chapter 3. The importance of providing SLA grammar lies in its role in unifying the structure of the SLA and standardising the vocabularies used to formally describe the offered/requested services. Furthermore, it is a first step towards providing a machine-readable SLA specification. In the machine-readable SLA specification, the data is structured following the proposed grammar. They can be processed by a computer and is presented in a CSV[1], JSON[2], XML[3], etc. format, with no need for human intervention in the interpretation. Having the SLA in a machine-readable format has advantages such as minimising the risk of confusion over the SLA interpretation [472, 290]. Additionally, it is an important step towards automating the processes of application deployment, monitoring and dynamic reconfiguration [472, 290]. Furthermore, providing a machine-readable end-to-end SLA for an IoT application enhances and automates the process of selecting a service provider and negotiating, monitoring, enforcing and managing the SLA [472, 290].

Several projects have focused on the development of SLA specification languages [347, 165, 77, 3, 445, 316] [4]. For example, [445] present a framework that enables application developers to specify SLA metrics, how they can be calculated, the evaluation period, and constraints to avoid SLA violations using their SLA grammar, termed XCLang. However, their main focus is the Cloud

[1] A CSV file (Comma Separated Values) is a plain text file that contains a data set, which is used for transferring data between different applications.

[2] JSON (JavaScript Object Notation) is a lightweight format for data exchange.

[3] Extensible Markup Language (XML) is a markup language that specifies a collection of document encoding rules in human-readable, machine-readable format.

[4] Refer to Chapter 2 for further details about these references

database tier.

Gámez Díaz et al [167] propose iAgree. iAgree is a language used to describe a vendor-neutral SLA, and it aims to model a considerable number of scenarios, including computational services (e.g., RESTful APIs) and human services (e.g., business processes).

The above-mentioned studies [227, 167, 347, 165, 77, 3, 445, 316] are works that define an SLA in a machine-readable format. However, none of the specification languages have been developed for the IoT. This means that there is no consideration of an end-to-end specification, which implies that the huge dependency between IoT layers has been neglected. In SLAs for IoT ecosystems, it is important to specify end-to-end contractual terms to ease the process of tracing when the quality of service becomes degraded [408]. Additionally, specifying end-to-end contractual terms ensures that service providers deliver services that match consumers' expectations and it protects their rights if they encounter any damage during the contract period [432, 248].

Therefore, the main goal of this chapter is to design and create an end-to-end SLA specification for the IoT while taking the following challenges into account [392]:

1. The multi-layered nature of the IoT (IoT device layer, Edge computing layer, Cloud computing layer).
2. Several metrics are required to capture the performance of the services and infrastructure resources for each layer of an IoT application (e.g., data freshness at the IoT devices layer and the latency of stream processing in the Cloud layer).

The main contributions of this chapter are as follows:

- Propose a new multi-layered grammar for the syntactical structure of an SLA specification for IoT applications.
- Evaluate the proposed grammar.

The remainder of this chapter is organised as follows: The SLA grammar is presented in Section 4.2. We evaluate our work and discuss the results in Section

4.3. Section 4.4 provides a comparison of approaches that are similar with respect to a number of important criteria. Our conclusions and future research directions are presented in Section 4.5.

4.2 SLA Grammar for IoT Applications

One of our main objectives is to provide a machine-readable SLA specification that can be used by an application orchestrator to automatically deploy IoT applications and monitor adherence to the QoS requirements. In this section, we propose an SLA grammar for IoT applications based on the conceptual model (presented in Chapter 3). We define the SLA grammar formally using the extended Backus-Naur form (EBNF). The EBNF is a context-free grammar that can define the syntactic structure of a language. A context-free grammar is a collection of recursive rules for creating string patterns and it consists mainly of [430]:

- Terminal symbols: These are the smallest block in the grammar (e.g., quoted literal and a regular expression); they can be defined as tokens and they are always on the right side of the production rule.
- Non-terminal symbols: These are defined by a set of terminals and other non-terminals and they are always placed on the left side of the production rules.
- Production rules: These are a series of production rules that replace non-terminal symbols. Production rules have the following form:
  non-terminal symbol $->$ non-terminal symbols and terminals symbols.

  Some of the operators used within the grammar are:

- "?" indicates that the symbol (or set of symbols in parentheses) to the left of the operator is/are optional and can be included or not included.
  To represent the number of occurrences/repetitions of a symbol, we use the following operators:
  - "*" means that the symbol (or set of symbols in parentheses) to the left can occur zero or more times.
  - "+" means that the symbol (or set of symbols in parentheses) to the left can occur one or more times.

The EBNF is used to define the syntactic structure of our proposed SLA specification for the IoT. Table 4.1 shows the grammar of the proposed SLA specification, which is formally defined in EBNF. The SLA grammar consists of a list of production rules; each production has a non-terminal symbol on its left-hand side, while its right-hand side represents the non-terminal production rule. The production rule consists of at least one non-terminal and/or terminal symbol. Terminal symbols are written between single or double quotation marks. The SLA language, then, can be produced from the given context-free grammar by simply producing a set of terminal symbols that result from frequently replacing any non-terminal in the sequence with its production rule. For example, in the first line of Table 4.1, the production rule for the non-terminal: $< SLA >$ is:
$< appType >? < slaId >? < startDate >? < endDate >? < description >? < slaState >?$
$< party > +? < slo >* < workflowActivity >* < budget >$
Furthermore, within the production rules, there are many non-terminal symbols. Each of the defined symbols in the production rules of non-terminal $< SLA >$ has its own production rule. Therefore, each symbol can be replaced by its production rule. For example, $< id >$ can be replaced by $< digit >$, and $< digit >$ can be replaced by its production rule:
$< digit >$:= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7'|'8' | '9'.
The process of replacing each non-terminal with its production rule is repeated until the outcome is an SLA specification language for the IoT.

Consider the following example: $< SLA >$ ::= $< appType >? < slaId >? < startDate >? < endDate >? < description >? < slaState >? < party > +? < slo >* < workflowActivity >* < budget >$
This example can be read as an SLA ($< SLA >$) that consists of optional characteristics id ($< slaId >$), a start date ($< startDate >$), an end date ($< endDate >$), a description ($< description >$), an SLA state ($< slaState >$) and a budget constraint ($< budget >$). This example additionally consists of one or more parties ($< party >$), zero or more SLOs ($< slo >$) and zero or more workflow activities ($< workflowActivity >$).

In the following, we give a brief description of each non-terminal symbol listed in the proposed grammar presented in Table 4.1. Some of the non-terminal symbol presented below are inspired by works presented in [472, 227, 256, 245, 476]

such as $< startDate > < endDate > < party >$. We introduce the non-terminal $< workflowActivity >$ symbol and its related non-terminal symbols listed in its production rule (e.g., $< activity > < service > < infrastructureResourceType > < configurationRequirement >$) as well as most of the listed domain-specific vocabularies (e.g., 'Persistence of Customer Information'| | 'Encryption Support'| 'No of vCPU' |..... | 'Write Capacity'):

4.2.1 <SLA>

An SLA attribute consists of the following:

- `<appType>` Indicates the type of application, such as a smart home application, remote health application, or smart metering.
- `<slaId>` Assigns a unique identifier to an SLA.
- `<description>` Assigns a descriptive context to the SLA.
- `<startDate>` Defines the start date of an agreement.
- `<endDate>` Defines the end date of an agreement.
- `<slaState>` Reflects the state of the SLA as an offer, request or finalised agreement. It helps in the SLA negotiation phase.
- `<party>` An attribute to hold attributes that describe the parties involved in an SLA. We describe the `<party>` parameter in more detail in 4.2.2.
- `<slo>` An attribute to define a list of the SLOs of a system (at the application level). For example, in the RHMS, one of the SLOs at the application level is "detect urgent cases within 300 seconds time unit". We describe the `<SLO>` attribute in more detail in 4.2.3.
- `<workflowActivity>` Lists the main workflow activities of the application. We describe the `<workflowActivity>` attribute in more detail in 4.2.4.
- `<budget>` : Specifies the financial cost/price limit of an SLA. It has the following elements:

Table 4.1 SLA Grammar of IoT Application

Non-terminal	Production rules
< *SLA* > ::=	< *appType* >? < *slaId* >? < *startDate* >? < *endDate* >? < *description* >? < *slaState* >? < *party* > +? < *slo* > + < *budget* > < *workflowActivity* > +
< *appType* >::=	'Smart Building' | 'Smart Traffic' |..| 'Smart City'
< *slaId* >::=	< *string* >
< *startDate* >::=	< *date* >
< *endDate* >::=	< *date* >
< *description* >::=	< *string* >
< *slaState* >::=	'offer' |'request'|'agreed On'
< *party* >::=	< *partyId* >+? < *name* >+ < *contact* >+ < *role* >+
< *partyId* >::=	< *digit* >
< *name* >::=	< *string* >
< *contact* >::=	< *string* >
< *role* >::=	'Cloud Provider' |'Network Provider' | 'Sensing Provider'|. . . | 'Broker' | 'IoT administrator' | 'End User'
< *budget* >::=	< *amount* > < *currency* >
< *amount* >::=	< *digit* >
< *currency* >::=	'$' |.....| '£'
< *slo* >:=	< *qosMetric* > < *priority* > < *requiredLevel* > < *value* > < *unit* > < *partyId* >? < *qosEvaluatingScheduale* >? < *action* >?
< *qosMetric* >::=	'Outage Length'|'Response Time' | 'Availability' |....| 'Timeliness' | 'Cost'
< *priority* >::=	'High'|'Medium'|'Low'
< *requiredLevel* >::=	'greater than' |'greater than or equal' | 'equal' | 'not equal' | 'less than' | 'less than or equal'
< *unit* >::=	'%' | 'millisecond' | 'seconds' |'minutes' | 'hour'| 'month' |'year' |'KB'|.....| ' per month'
< *workflowActivity* >::=	< *activity* > < *service* >* < *infrastructureResource* >*
< *activity* >::=	'Capture event of interest(EoI)' | 'Examine the captured (EoI) on fly' |..| 'Store Unstructured Data'
< *service* >::=	< *serviceType* > < *slo* > ∗ < *configurationRequirement* > ∗ < *partyId* > ∗ < *price* >?
< *serviceType* >::=	'sensingService' | 'batchProcessingService' |.....| 'machineLearningService'
< *infrastructureResource* >::=	< *infrastructureResourceType* > < *slo* > ∗ < *configurationRequirement* > ∗ < *partyId* > ∗ < *price* >?
< *infrastructureResourceType* >::=	'IoTDevice' | 'CloudResource' | 'EdgeResource'
< *configurationRequirement* >::=	< *booleanBasedConfiguration* > | < *typeBasedConfiguration* > | < *numericalBasedConfiguration* >
< *booleanBaseConfiguration* >::=	< *configurationFeature* > < *value* >
< *configurationFeature* >::=	'Persistence of Customer Information'| | 'Encryption Support'| 'No of vCPU' |..... | 'Write Capacity'
< *typeBasedConfiguration* >::=	< *configurationFeature* > < *type* >
< *type* >::=	'SSD (local machine)'|..... | 'HDD (local machine)'
< *numericalBasedConfiguration* >::=	< *configurationFeature* > < *requiredLevel* > < *value* > < *unit* >?
< *qosEvaluatingScheduale* >::=	< *startAt* > < *unitBase* >
< *startAt* >::=	< *time* > < *date* >
< *unitBase* >::=	'hourly' 'daily' |....|'monthly'
< *action* >::=	'Send Notification' |....| 'Refund As Credit'
< *price* >::=	< *amount* > < *currency* > < *perUnit* >
< *perUnit* >::=	'per data size' |....| 'per VM type'
< *value* >::=	< *string* >| < *digit* > | < *double* > | < *Boolean* >

  - <amount>: Specifies the amount of money needed to pay for the service under particular requirements. For example, in the SLA clause of the RHMS, subscribers pay 100 dollars to use the service.
  - <currency>: Specifies the currency (e.g., dollars).

4.2.2 <Party>

This attribute specifies the parties involved in an SLA [165]. It has the following elements:

- <partyId>: The unique identification of a party involved in an SLA.
- <name>: Specifies, textually, the name of the party.
- <contact>: Specifies the contact details of a party. The contact details can include phone number, email address and home address.
- <role>: Specifies, textually, the expected role of a party. For example, the role of a network provider is “providing networking service”.

4.2.3 <slo>

This is an attribute of an <slo>. It defines the metric of interest to measure the performance of a system with regard to the SLO requirements. An SLO could minimise the latency to be less than x time unit; latency in this SLO clause is the QoS metric of the *SLO*.

- <qosMetric>: Used to name a QoS of interest, for instance, the <qosMetric> of an *slo* is latency.
- <priority>: Specifies the priority of the SLO based on consumer preferences [421]. Each SLO has a priority level: high, median or low. Typically, the priority attribute is used if there is a need to trade off between two or more SLOs. It is also considered for resource provisioning and traffic control purposes.
- <requiredLevel>: Defines operators that are part of the expression. The required level could be greater than, less than, or less than or equal to. For instance, the latency of *SLO* should be less than 300 seconds.

- <value>:Specifies a threshold value of a QoS metric. For example, in *SLO*, the latency should be less than 300 seconds, and the threshold value is 300.

- <unit>: Specifies a unit of a threshold value. For instance, where the time constraints in *SLO* should be less than 300 seconds, the unit value is seconds.

- <partyId>: Specifies the ID of the party who is responsible for guaranteeing the SLO. For example, the provider of the RHMS is the <partyId> who is in charge of providing the service to detect urgent cases among subscribed patients while respecting the agreed-upon time constraints.

- <qosEvaluatingSchedual> This defines the schedule for evaluating the QoS requirement. In other words, it checks whether the service is running at the expected level. For example, the SLA clause of the RHMS specifies that every day at 9:00 am, the statistics of the required QoS metric, such as "required time for detecting urgent cases", are evaluated. The evaluation can be performed daily, in which case the evaluation period is between the last scheduled check (for example, 9:00 am, 11 May 2017) and the next scheduled check (9:00 am, 12 May 2017).
  <qosEvaluatingSchedual> has the following attributes:

  - <startAt>: This is a date and time format that indicates when the required metrics are scheduled to be evaluated against the required threshold value – from the SLA clause of the RHMS, startAt it is "9:00 am 12 may 2017".
  - <unitBase>: Expresses the intervals at which the validation should be performed on the basis of "minutely", "hourly", "daily", "monthly", and "yearly". The SLA clause of the RHMS indicates that the <unitBase> is "daily".

- <action> This parameter specifies an action, such as send notification, apply reconfiguration policy, or apply compensation policy, that should be taken if there is a violation of an SLO constraint.

4.2.4 <workflowActivity>

This parameter lists common activity and consists of the following elements:

- <activity> describes the activity in text, such as: “capture event of interest” or “store unstructured data”.
- <service> describes the service required to accomplish the activity. It has the following elements:
  - <serviceType>: It includes services such as “sensing service” and “networking service”.
  - <slo>: This element is same as that described in 4.2.3.
  - <configurationRequirement>: This element is described in 4.2.5
  - <partyId>: This element refers to the *Party* providing the service.
  - <price>: This element is described in 4.2.6
- <infrastructureResource> describes the resources required to host the service that is needed to accomplish the activity. It has the following elements:
  - <infrastructureResourceType> includes infrastructure resource types such as “IoT device”, “Cloud Resource” and “Edge Resource”.
  - <slo> has the same element as described in 4.2.3.
  - <configurationRequirement> is described in 4.2.5.
  - <partyId> refers to the *Party* providing the service.
  - <price> is described in 4.2.6.

4.2.5 <configurationRequirement>

specifies the requirements related to some configuration parameters of the associated infrastructure resource and/or service. It can be one of the following types:

- <booleanBasedConfiguration> specifies the configuration parameter that has a Boolean value. It consists of the following elements:

  - <configurationFeature>: To reflect the feature, we seek to specify its value, such as “compression support” to compress data and the “encryption support” feature.
  - <value>: To reflect the value assigned to “configurationFeature”, for example, to specify that the ingestion service should support data compression, within the block that specifies the configuration requirements of the ingestion service, we can assign the following values: configurationFeature: “compression Support”, value: “true”,

- <typeBasedConfiguration> is used to reflect the specification of a feature that has a type-based value, such as a type of cluster, and it consists of the following elements:
  - <configurationFeature>: Reflects the feature of which we seek to specify the value, such as "type of cluster"
  - <type>: Reflects the value assigned to “configurationFeature”, for example, to specify that the batch-processing service requires a map-reduce cluster. Therefore, within the block that specifies the configuration requirements of the batch-processing service, we can assign the following values: configurationFeature: “type of cluster”, type: “map-reduce”,

- <numericalBasedConfiguration> is used to describe the configuration requirement that requires a numerical value and it consists of the following elements:
  - <configurationFeature>: To reflect the feature, we seek to specify its value.
  - <requiredLevel> defines operators that are part of the expression. The required level could be greater than, less than, or less than or equal to.
  - <value> reflects the actual numerical value.
  - <unit> reflects the unit. For example, the above-mentioned elements of a <numericalBasedConfiguration> can be used to describe a configuration requirement of the store service: configurationFeature: “read

capacity”, requiredLevel: “greater than” value: "50", unit:"tuples per second".

4.2.6 <price>

This is a parameter to specify the financial cost/price of subscribing to a service. It has the following attributes:

- <Amount> specifies the amount of money to pay for the service under particular requirements. For example, in the SLA clause of the RHMS, subscribers pay 100 dollars to use the service.
- <Currency> specifies the currency (e.g., dollars).
- <PerUnit> specifies the base for payment based on the size of the sent data, per response, or per month of subscription. In the SLA clause of the RHMS, payment is per month.

The proposed grammar allows the specification of an SLA at the application level to be used between the consumer and service provider at the front end. Additionally, it allows each required service to be specified at a fine-grained level of detail. Therefore, we believe that an SLA specification based on the proposed grammar can be specified on an end-to-end basis, and it can then be used by system engineers as well. In our grammar, we use the "workflow activities" concept to contain the list of involved activities – for example, in the RHMS use-case, "collect patient’s data" matches the "capture event of interest" activity in our grammar. Each activity is associated with a service (or services) and an infrastructure resource to deploy the service(s). Both the service and the infrastructure resource have their own SLO constraints as well as configuration requirements (see Figure 4.1).

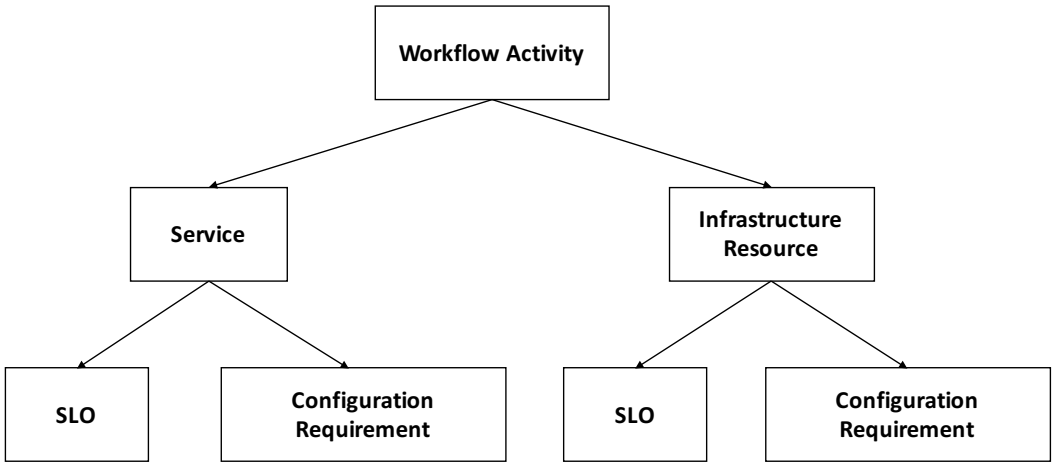

Fig. 4.1 Conceptual mapping to reflect the relationship between workflow activity and service and infrastructure resource concepts

One of the advantages of using a free-context grammar is the ability to reduce misunderstandings by providing only one interpretation. The elements $< appType >? < slaId >? < startDate >? < endDate >? < description >? < slaState >?$ and $< party > +?$ describe basic information related to the SLA. Each SLA consists of at least one service level objective $< slo >$ to express the required QoS at the application level (e.g., in the RHMS, `response Time` is less than 2 minutes). Each $< slo >$ has a priority level (e.g., high, medium, low). For example, in the RHMS, response time has a higher priority than power consumption; in contrast, for an auto-illuminated building, power consumption has a high priority. The concept of a $< workflowactivity >$ is used to express the data flow activities of an IoT application (e.g., capture the event of interest and perform small or large-scale real-time data analysis and large-scale historical data analysis). Each workflow activity is mapped to its required $< Service >$ (such as sensing service, batch-processing service) and to its $< InfrastructureResource >$ (e.g., IoT devices, Edge resources, Cloud resources). Each service and infrastructure resource has its own $< slo >$ and $< configurationMetrics >$. As mentioned above, the SLO can express the required QoS for each of the services. Configuration requirements such as $< numericalBasedConfiguration >$, $< booleanBasedConfiguration >$ and $< typeBasedConfiguration >$ are differentiated based on their values: some configuration features have Boolean values, others determine the type of feature, and some have numerical values. For example, the number of required CPUs, encryption support and type of cluster are examples

of $< numericalBasedConfiguration >$, $< booleanBasedConfiguration >$ and $< typeBasedConfiguration >$, respectively.

4.3 Evaluation

After identifying the reference architecture, the main concepts to be considered within the SLA and the related vocabularies, and after proposing a grammar for SLA specification, this section presents our approach to evaluating the proposed SLA specification for IoT applications. In Section 4.3.1, we introduce the goal/question/metric (GQM) approach [98]; then, we apply the GQM approach to serve our purpose in Section 4.3.2. We present a discussion in Section 4.3.2.

4.3.1 Goal/question/metric (GQM) approach

A software system's success can be measured by the extent to which it meets its intended purpose. The process of identifying and documenting stakeholders and their needs, which is conducive to analysis, communication and subsequent implementation, can be defined as software system requirement engineering [364] [386]. The activities of requirement engineering are designed to manage all knowledge related to requirements. Such knowledge is commonly reflected in a number of objects, such as use cases, storyboards, natural language documents and specifications of business processes, which are called system artefacts [386].

Goal modelling is a key part of requirement engineering activities. Goal models identify stakeholders and business goals, alternative ways to meet the goals and the positive/negative impact of the goals on various quality aspects. The analysis of these models guides decisions and the refinement of inaccurate user requirements towards accurate system requirements [35]. Information collection can be based on explicit or implicit approaches [68]. In the explicit approach, information is collected directly from the user, usually through web-enrolment forms, surveys or psychometric instruments designed for a specific purpose. On the other hand, in the implicit approach, the system automatically extracts implicit information, such as tracking user behaviour [68].

Many mechanisms have been introduced in the literature to define measurable goals [98], such as software quality metrics (SQM) [82] [320], the goal/question/-metrics (GQM) approach [64] [65] [66] [67] and the quality function deployment (QFD) [240] approach. The GQM approach combines the majority of the current measurement approaches and generalises them to include processes, resources, and products. The GQM approach is an adaptable approach and it can be applied in different environments; it has been adopted by a number of institutions, e.g., NASA, Hewlett-Packard [179], Motorola, and Coopers & Lybrand.

The GQM approach specifies a number of steps to be undertaken to determine whether the goals have been achieved [98]. First, the goals must be clearly specified; then, a path must be traced between these goals and the data that define them. These data can then be interpreted through a framework against the predetermined goals. Quantified information can be used to measure whether the goals have been achieved [98].

The approach was initially used to assess weaknesses in a set of projects in the NASA Goddard Space Flight Center environment. Although the approach was initially utilised to characterise and assess objectives for a specific extension in a specific environment, it has since been employed to define and assist goals for a certain project within a certain environment, such as the objective-setting step in an evolutionary quality-improvement model customised for a software development organisation. The result of the application of the GQM approach is a measurement framework focusing on a specific set of issues and a set of rules to interpret the measured data. The measurement framework has three levels [98]:

1. Conceptual level (GOAL): At this level, a goal is defined for an object for different models of quality within a particular environment and can be from different points of view for a variety of reasons.

2. Operational level (QUESTION): A set of questions attempts to characterise the object of measurement (product, process, resource) with regard to a chosen quality issue and to determine, as a result, its quality from that point of view.

3. Quantitative level (METRIC): This refers to quantifying the answer to a question by associating a set of data with each question.

A GQM framework consists of a hierarchical structure (Figure 4.2), starting with a goal. Then, the goal is refined into a set of questions to break the issue to be measured (defined within the goal) down into its key components. Each question is refined into metrics as a step towards quantifying the answers to the questions.

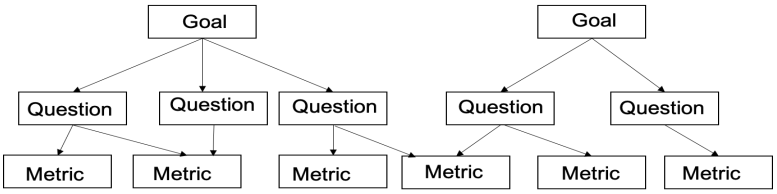

Fig. 4.2 The hierarchical structure of a GQM model [98]

4.3.2 Applying the GQM approach to evaluate the Proposed SLA Specification for IoT Applications

The GQM approach has been applied in different studies (such as those presented in [6, 127, 503, 11]), and it has shown its effectiveness in serving the purposes for which it has been applied. Achtaich et al [6] apply the GQM approach to assess the expressiveness, domain independence and scalability of the state-constraint transition (SCT) modelling language. SCT is a language that extends the finite state machine (FSM) [195] paradigm to describe the dynamic behaviour of self-adaptive systems. Darweesh et al [127] provide a general approach to determine an agent's security in multi-agent system environments based on a GQM approach. The performance of the proposed framework is measured as the percentage of fulfilment of a set of security requirements, such as confidentiality, authentication, repudiation and access control. Yahya et al [503] apply a GQM approach to construct security metrics, which evaluate the security control features, and once these metrics are defined, they can be applied to evaluate the Cloud storage security of an organisation.

We believe that applying the GQM approach allows us to determine whether or not we have achieved our intended goal of proposing a grammar, especially

because it provides a roadmap that we can use to measure the desired goals numerically. The GQM approach was chosen because there is a similarity between what we want to achieve and the GQM approach. We intend to check the generalizability and expressiveness of the proposed specification, which can be expressed as the GOAL in the GQM approach. Under each goal, there are a number of concepts that we can formulate as questions (corresponding to the questions in the GQM approach), and we can calculate the metric value for each question based on the participants' answers (corresponding to the METRIC in the GQM approach). In the following subsection, we follow the GQM approach to define our goals and present the list of questions that will be used to calculate the metrics and reflect the percentage achieved for each stated goal.

Defining the Goals

We aim to measure the generalizability and the expressiveness level of our proposed grammar from the viewpoints of IoT experts/IoT administrators. Therefore, we intend to specify each of the issues (generalizability and expressiveness of the proposed specification) as the GOAL based on the GQM approach. The GQM method provides a template to define a GOAL unambiguously by expressing the following main elements: purpose, perspective and context characteristics (Table 4.2 illustrates the main elements of the GQM goal definition template [479]).

Table 4.2 Main elements of the GQM goal definition template [479]

Analyse	the object under measurement
for the purpose of	understanding, controlling or improving the object
with respect to	the quality of the object on which the measurement focuses
from the viewpoint of	the people who measure the object
in the context of	the environment in which the measurement takes place

We set two separate goals (see Table 4.3 and Table 4.4), following the template for designing a goal as defined in the GQM approach [98]:

Table 4.3 Defining our first goal following the template in [98]

Goal 1:	Main Element from [98]	Example related to our work
	Purpose:	indicate
	Issue:	the generalizability
	of object:	of the proposed grammar
	viewpoint:	from the IoT experts′/IoT administrators′ viewpoints

Table 4.4 Defining our second goal following the template in [98]

Goal 2:	Main element from [98]	Example related to our work
	Purpose:	indicate
	Issue:	the expressiveness level
	of object:	of the proposed grammar
	viewpoint:	from the IoT experts′/IoT administrators′ viewpoints

Defining the Questions

For each of the predefined goals, a list of questions is prepared. The answers to the questions are quantified to measure whether or not the goal had been achieved. The main purpose of the prepared questions is to check whether the vocabulary terms that we consider within the grammar can capture the requirements for different use-case scenarios from the viewpoints of the IoT experts/IoT administrators. Furthermore, using free-text types of questions, the participants have an opportunity to express what other considerations they believe to be important.

Defining the Metric

For each question, we tried to quantify the answer by calculating the users' satisfaction regarding whether or not the grammar could capture the requirements for different use-case scenarios and what else should be considered for different parts of the concepts within the grammar. To calculate the satisfaction percentage for each goal, the following steps were applied:

- Count the number of missing requirements (NMR). The NMR represents the number of requirements that we failed to consider from the participants'

viewpoints. Additionally, count the number of selected requirements (NSR) from the predefined list. The NSR is the number of requirements that a participant selected from the predefined list. Then, divide the NMR by the NSR from the predefined list (see equation 4.1). The result of equation 4.1 reflects the ratio of what is missing with regard to question$_i$ from participant$'_j s$ point of view. The miss ratio from participant j's point of view can be calculated as follows:

$$miss\_ratio_j = \frac{NMR}{NSR} \tag{4.1}$$

To calculate the metric value for each question, we apply equation 4.2 to each question$_i$:

$$metric_i = \frac{\sum_{j=1}^{j=P} miss\_ratio_j}{p} \tag{4.2}$$

i represents the question for which we are interested in calculating the metric value. $j = 1 \cdots p$, where p represents the number of participants.

- Calculate the overall value, which represents the satisfaction percentage of achieving the kth goal by applying the following 2 steps:

  1. The overall value, which represents the unsatisfied percentage of achieving goal$_k$:

  $$unsatisfiedPercentage_{goal_k} = \frac{\sum_{i=1}^{i=n} Metric_i}{n} * 100 \tag{4.3}$$

  $i = 1 \cdots n$, where n represents the number of predefined questions for goal$_k$. $k = 1 \cdots G$, where G represents the number of predefined goals.

  2. The overall value, which represents the percentage of satisfying goal$_k$, is calculated as follows:

  $$satisfiedPercentage_{goal_k} = (1 - \frac{\sum_{i=1}^{i=n} Metric_i}{n}) * 100 \tag{4.4}$$

  $i = 1 \cdots n$, where n represents the total number of predefined questions for goal$_k$
  $k = 1 \cdots G$, where G is the total number of predefined goals.

Experiment

The main purpose of this experiment is to evaluate our proposed grammar and determine whether or not it met the predefined goals: generalizability and expressiveness (i.e., capturing the users' requirements). The potential users of our proposed work are IoT administrators . Therefore, we conducted the experiment with participants whose research interests were mainly related to the IoT, Cloud computing, Edge computing and networking.

Procedure

The experiment was conducted following a well-defined procedure and there were 11 participants. First, a focus group discussion was held in which the participants received an introduction to the SLA and our reference architecture for the IoT, the conceptual model, and the grammar. The number of members per focus group varied depending on the availability of the participants. The participants were allowed to discuss, ask for explanations and provide instant suggestions. A use-case was introduced for clarification purposes. At the end of the focus group, the participants were asked to fill out a paper-and-pen version of a questionnaire in which there were three questions related to the first goal (indicate the generalizability of the proposed grammar from IoT experts'/IoT administrators' viewpoint) and eleven questions related to the second goal (indicate the expressiveness of the proposed grammar from IoT experts'/IoT administrators' viewpoint). Table 4.5 sheds light, briefly, on the participants' research interests.

Reducing Bias

Considering that the questionnaire is, by its nature, vulnerable to bias, the researcher has taken the following measures as far as possible to mitigate the risk of bias:

- The participants were informed that all the collected data would be confidential and would not be shared with others. The privacy and anonymity of participants were preserved. All the data obtained will only be used for evaluation by the researcher and the supervisor. This ensured that the participants gave their feedback with no external influence.

- The participants were not subject to any influence or time constraints. The researcher conducted the introductory workshop and replied to any questions the participants posed. Each participant gave his/her feedback separately at the end of the session to prevent any external influence (e.g., influence of colleagues). Furthermore, their name was not required when answering the questionnaire for anonymity purposes and to seek an honest opinion without any influence.

- A number of participants had participated before in Chapter 3. However, to reduce the impact of the possibility that those participants would apply the same approach when answering the questions, or have the same understanding, the nature of the questions was different from those presented in the previous chapter (i.e., Chapter 3). Furthermore, the participants in this study were mixed – some had participated before, but for the rest this was their first time participating.

Table 4.5 Participants' research interest

Participant	**Research Interest**
1	IoT researcher interested in SLAs in the context of the IoT using Blockchain
2	IoT, Cloud computing and networking
3	IoT fault tolerance
4	IoT data management and analytics
5	IoT security mechanism using Edge infrastructure, network security between sensor and Edge and between Edge and Cloud privacy on low resource devices
6	Interested in designing a scalable data stream processing system within the IoT paradigm. His project focuses on automating computational placement in IoT systems, pushing the computation as close to the data source as possible, considering a range of non-functional requirements such as energy and bandwidth
7	Conducting research on IoT data management, mainly focusing on IoT data Google discovery and retrieval, also applying data stream processing techniques in early walking system applications
8	Interested in the IoT since he is working on a project that aims to make a real-time ambulance system. The project also considers batch processing.
9	Researcher of IoT projects
10	Security of the IoT
11	Research related to IoT and Cloud projects

Experiment results

For each of the predefined goals, a list of questions was prepared (3 questions for Goal 1 and 11 questions for Goal 2). For each question, there was a checklist of requirements (see Figure 4.3 as an example). The participants were asked to check the requirements that, from their experience, they thought it is good that we considered, and to write down the missing requirements that they thought we should add.

The column headers of Table 4.6 represent the question number and the count of the vocabularies that were defined to capture the requirements for the concept associated with each question[5]. For example, the second column header is Q1/10, which means that 10 elements were defined that were related to the concept (workflow activity). (Figure 4.3 shows question 1, which is related to the workflow activity concept).

Question 1: Considering the following predefined list of workflow activities

- ☐ Capture event of interest (EoI)
- ☐ Examine the Captured EoI on fly
- ☐ Filter Captured EoI
- ☐ Aggregate the Captured EoI
- ☐ Ingest Data from one or more data resources
- ☐ Large-Scale Real-time data analysis
- ☐ Large-Scale Historical data analysis
- ☐ Apply machine learning approach
- ☐ Store Structured Data
- ☐ Store Unstructured Data

1. Does the predefined list of workflow activities cover your IoT project's workflow activities?
   Answer: Yes or No .....
2. Could you please "tick" the activities that you believe will be involved/part of your application/project.
3. Do you think more workflow activities should be included?

   Answer: Yes or no .....
4. if your answer in 3 is "yes", could you please list the workflow activities that you suggest considering

.....

Fig. 4.3 Sample of the questions given to the participants

From the participants' answers, we calculated the metric for each question and then calculated the satisfaction percentages for Goal 1 and Goal 2. We followed the following steps:

1. Step 1: Check the answer to each question, by checking whether the participant thought that the predefined list of requirements covered his/her

[5]Refer to Appendix A for the full questionnaire

Table 4.6 Number of selected vocabularies for each question

Participant-ID	Q1/10	Q2/3	Q3/8	Q4/8	Q5/13	Q6/20	Q7/4	Q8/10	Q9/12	Q10/5	Q11/18	Q12/15	Q13/11	Q14/12
1	6	2	4	4	8	7	1	4	6	2	8	0	3	3
2	9	3	5	6	13	16	3	9	9	5	7	8	5	2
3	10	2	4	8	11	15	4	10	4	4	7	10	6	7
4	9	2	6	8	10	16	4	10	9	4	16	13	6	11
5	8	3	7	8	13	20	4	10	12	5	18	15	11	12
6	6	3	5	5	9	12	4	9	8	0	13	9	8	9
7	10	3	8	8	13	20	4	10	12	5	18	15	11	12
8	10	3	8	8	13	20	4	10	12	5	18	14	11	12
9	7	3	7	6	12	4	4	6	4	5	14	8	7	6
10	6	2	7	7	10	19	4	8	11	4	13	11	10	11
11	10	3	8	6	13	20	4	10	0	5	13	14	8	9

IoT project's requirements. For example, for the question reflected in Figure 4.3 , we checked whether the participant thought that the predefined list of workflow activities covered his/her IoT project's workflow activities. We counted the NSR that the participants agreed/believed should be considered. For example, participant 2 selected 9 activities from the predefined list when he/she answered the first question (Figure 4.3, which asked whether he/she thought that the predefined list of workflow activities covered his/her IoT project's workflow activities). Table 4.6 shows the NSR for each question.

2. Step 2: Count the NMR that the participants suggested regarding the concepts they were asked about. For example, the second participant suggested one additional activity that he/she believed should be considered in the answer to "Could you please list the workflow activities that you suggest should be considered?" (column 2 of Table 4.7 shows the NMR for question 1, which is related to workflow activity).

3. Step 3: Calculate the metric: Based on the participants' answers to each question, we calculated the corresponding metric for each question following equation 1. For example, of 10 predefined lists of activities, column 2 of Table 4.7 shows how many activities were selected by the 11 participants and column 3 shows how many additional activities were suggested. Based on each participant's answer, we calculated the ratio of the activities missing from the predefined list following equation 1 in column 4 of Table 4.7 for each participant. For example, the second participant selected 9 activities from the predefined list and suggested one additional activity that he/she believed should be considered. Based on his/her answers, we calculated the miss ratio for this participant for question 1 as NMR/NSR=1/9. Then, we

calculated the corresponding metric value for question 1 following equation 2: Metric1=0.0554.

Table 4.7 Participants' responses to question 1 as a first step to calculating the metric value of question 1 (Q1)

Participant-ID	Selected Activities for Q1 /10	Missing/suggested Activities for Q1	Miss ratio applying equation 4.1
1	6	0	0
2	9	1	0.11
3	10	0	0
4	9	0	0
5	8	0	0
6	6	1	0.16
7	10	1	0.1
8	10	1	0.1
9	7	1	0.14
10	6	0	0
11	10	0	0

4. Step 4: After calculating each question's metric, for each predefined goal, we calculated the overall value, which represents the percentage of achieving the goal. We calculated the average value of all the metrics that represented the numerical value of the questions related to each goal following equation 4.3. For example, to calculate the overall value of the percentage achieved of goal 1, there were three questions. Therefore, we calculated the average value based on the calculated metric for each question related to goal 1 (metrics of questions Q1, Q2, Q3). Hence, the dissatisfaction percentage for achieving goal 1 was 8.30% (applying equation 4.3), while the satisfaction percentage for achieving goal 1 was 91.70% (applying equation 4.4). Table 4.8 and Table 4.9 reflect the calculated metrics, the overall dissatisfaction percentages and the overall satisfaction percentages for Goals 1 and 2.

Table 4.8 Calculated metrics, overall dissatisfaction percentage and overall satisfaction percentage of goal 1

Participant-ID	**Q1**	**Q2**	Q3
1	0.00	0.00	0.25
2	0.11	0.00	0.60
3	0.00	0.50	0.00
4	0.00	0.00	0.17
5	0.00	0.00	0.14
6	0.17	0.00	0.00
7	0.10	0.33	0.13
8	0.10	0.00	0.00
9	0.14	0.00	0.00
10	0.00	0.00	0.00
11	0.00	0.00	0.00
Metric of each Question	5.64%	7.58%	11.68%
The overall dissatisfaction percentage of goal1 :			**8.30**%
The overall satisfaction percentage of goal1 :			**91.70**%

Table 4.9 Calculated metrics, overall dissatisfaction percentage and overall satisfaction percentage of goal 2

Participant-ID	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14
1	0.25	0.00	0.00	1.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00
2	0.50	0.31	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.50	0.08	0.05	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.05	0.00	0.00	0.00	0.40	0.00	0.07	0.00	0.00
8	0.25	0.00	0.00	0.50	0.00	0.00	0.00	0.06	0.00	0.09	0.00
9	0.67	0.08	0.00	0.25	0.00	0.00	0.00	0.07	0.00	0.14	0.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.36	0.00	0.00
11	0.00	0.08	0.00	0.25	0.10	0.00	0.00	0.00	0.00	0.00	0.00
Metric of each Question	19.70%	4.95%	0.91%	23.48%	1.92%	0.00%	14.09%	1.15%	3.91%	2.13%	0.00%
The overall dissatisfaction percentage of goal2:							6.57%				
The overall satisfaction percentage of goal2:							93.43%				

Evaluation Analysis

We encountered several problems when trying to evaluate the proposed SLA specification, such as the complexity of the domain and the lack of available guidelines. Therefore, we evaluated the proposed SLA specification by reviewing it and discussing it with a considerable number of IoT experts. We applied the GQM approach to determine the generalizability and the expressiveness of the considered vocabularies within the proposed grammar for an SLA specification

for the IoT. The results indicate a high level of satisfaction for both goal generalizability and expressiveness.

Figure 4.4 displays the results based on the participants' satisfaction with the generalizability of our proposed grammar. For example, the generalizability of the predefined list of workflow activities and the considered computing layers is 94.36% and 92.42%, respectively, while the generalizability of the predefined services that were considered is 88.32%.

Figure 4.5 presents the results of the participants' satisfaction with the expressiveness of our proposed grammar. The expressiveness of the considered vocabularies in terms of capturing user requirements for different concepts is scored as follows: the percentage values of the expressiveness of the vocabularies for sensing, networking and ingestion services are 80.3%, 95.05% and 99.09%, respectively. It seems that there is a high level of satisfaction with the activities that were considered for the networking and ingestion services, while there is a lower level of satisfaction with the vocabularies that were considered to express user requirements for the sensing service. Some participants suggested the following vocabularies to capture user requirements for the sensing service: location of device, data generation rate, type of data generated (temperature, humidity), and, for IoT devices, processing capabilities (CPU speed, memory size).

Regarding the expressiveness of the vocabularies in terms of capturing the requirements of sensing, networking and ingestion services, the percentage values were 76.52%, 98.08% and 100%, respectively. There is a higher level of satisfaction with the activities that were considered for networking and ingestion services than those that were considered for the sensing service. Regarding machine learning, stream processing, batch processing, SQL database, and NoSQL database services, the percentage values of the expressiveness of the vocabularies used for each are 85.91%, 98.85%, 96.09%, 97.87% and 100%, respectively. There is a lower level of satisfaction with the vocabularies that were considered for machine learning. Some participants suggested the following vocabularies to capture user requirements regarding machine learning: type of machine learning classification, features for training and prediction for real-time data or batches,

and feature extractor.

Table 4.8 and Table 4.9 present the metrics calculated for each question based on each participant's answer. They also present the overall percentages for each goal, showing to what degree, in percentages, we achieved our goals. We are striving to achieve above 75% for each goal. As depicted in Table 4.8 and Table 4.9, we achieved 91.70% of our goal of providing a general grammar for different IoT applications and 93.43% of our goal of providing an expressive grammar to capture user requirements.

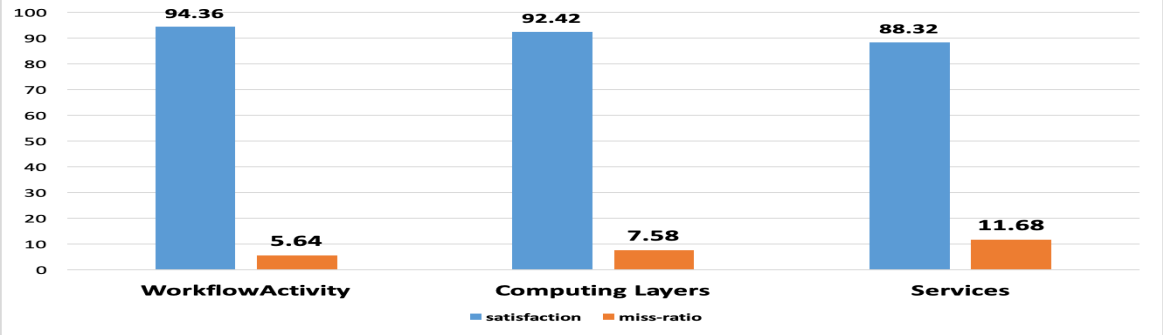

Fig. 4.4 Satisfaction ratio and miss ratio for all questions related to goal 1 "Generalizability of the grammar"

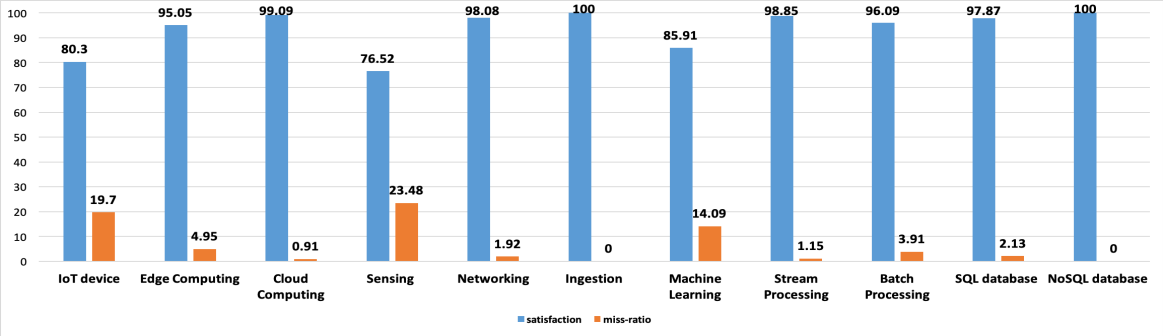

Fig. 4.5 Satisfaction ratio and miss ratio for all questions related to goal 2 "Expressiveness of the grammar"

For the missing requirements, the next question was the suggested workflow activities. As indicated in the chart below (Figure 4.6), just more than half of the respondents (55%) reported that the mentioned activities were quite sufficient, (27%) of the respondents suggested adding activities about notifications of unexpected events, (9%) of the respondents suggested adding activities on data

transformation, and (9%) suggested ML for training the module. The majority of respondents were quite happy with the current workflow activities.

Fig. 4.6 Suggested Workflow Activities

The next question was about the suggestion regarding computing layers. The results show that the mentioned computing layers were sufficient according to (82%) of the respondents, while (18%) of them suggested adding network and communication layers (see Figure 4.7 below). These results indicate that the computing layers are covered well in this questionnaire.

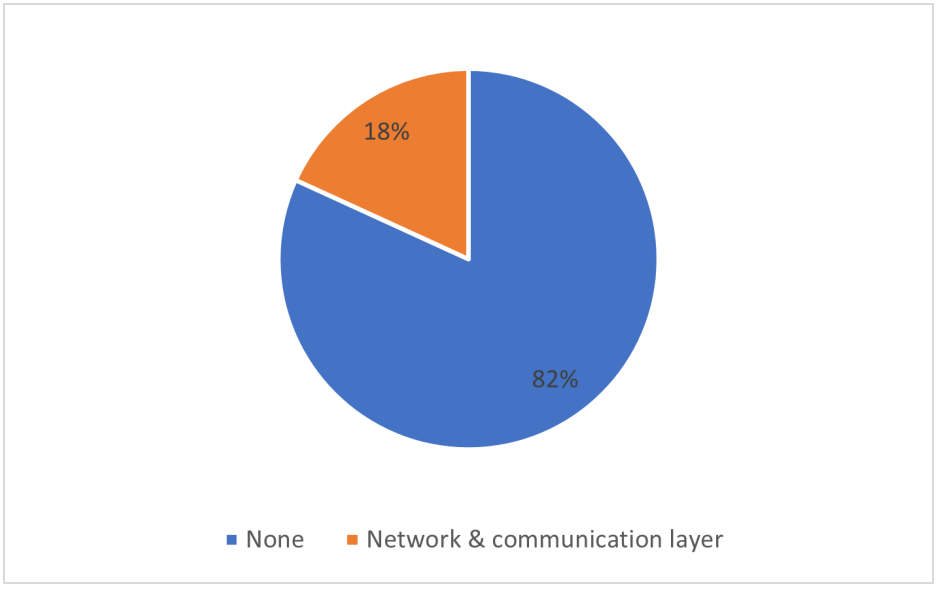

Fig. 4.7 Suggested Computing Layers

Then there was a question about the suggested services. As indicated in the figure below (Figure 4.8), just more than half of the participants (55%)

had no suggestions for additional services, (18%) of them suggested adding actuation service, others suggested adding management and configuration (9%), monitoring service (9%), and messaging service (9%). This means that there is a need to expand services.

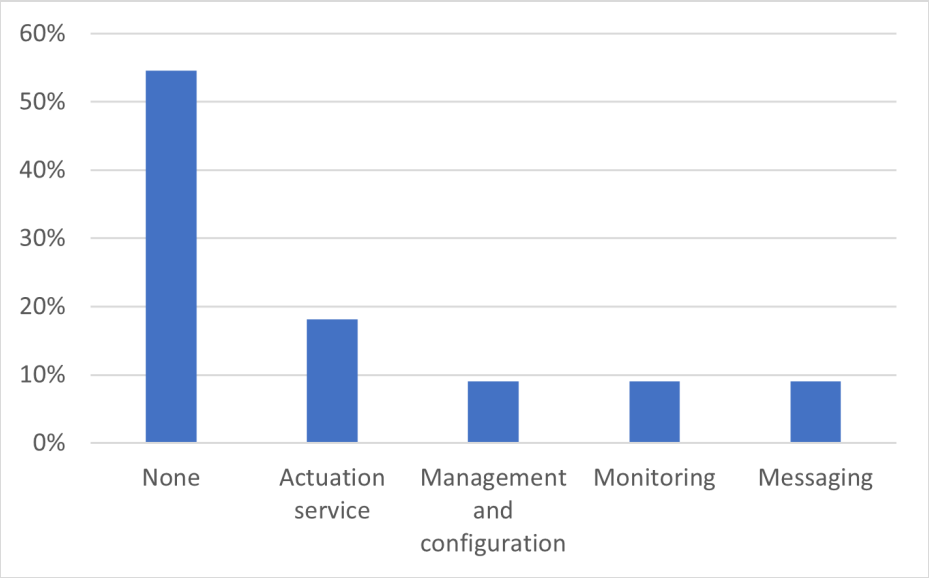

Fig. 4.8 Suggested Services

Then the participants were asked if they had any suggestions regarding consumer requirements for IoT devices. As is obvious in the figure below (Figure 4.9), (43%) of the respondents had no suggestions, (29%) of them suggested vocabularies to express the capabilities of devices, (14%) of them suggested the type and rates of data generation, (7%) required an addition in the form of privacy policies, and the same percentage required messaging protocols. Therefore, the listed consumer requirements should be taken into consideration.

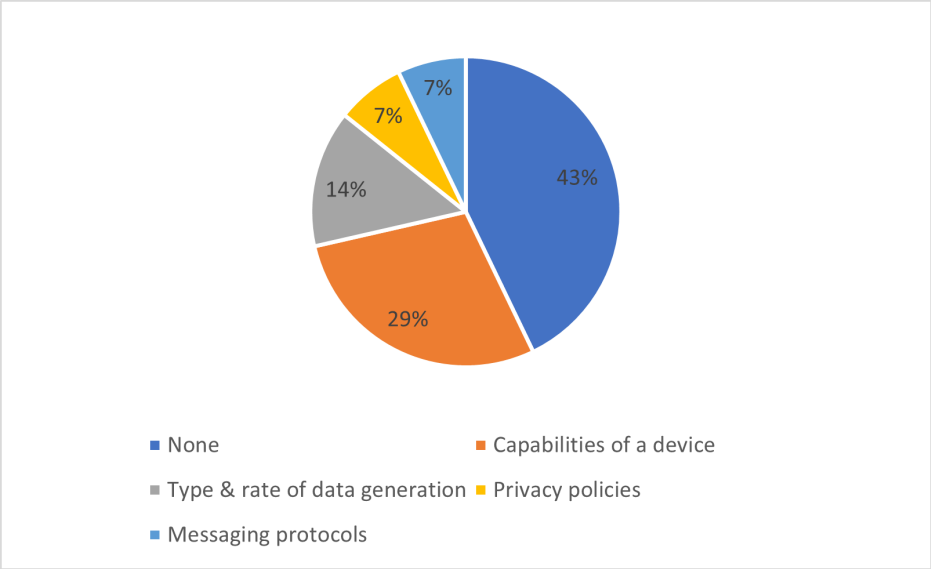

Fig. 4.9 Suggested Requirements for IoT devices

The next question was about the list of requirements for Edge computing. As indicated in the chart below (Figure 4.10), the majority of the participants (64%) were satisfied with the requirements that already existed in the questionnaire. However, some participants had their own requirements: (18%) operating system, (9%) device location and (9%) reliability. Thus, the list of SLO metrics and the configuration requirements of sensing services need some further additions.

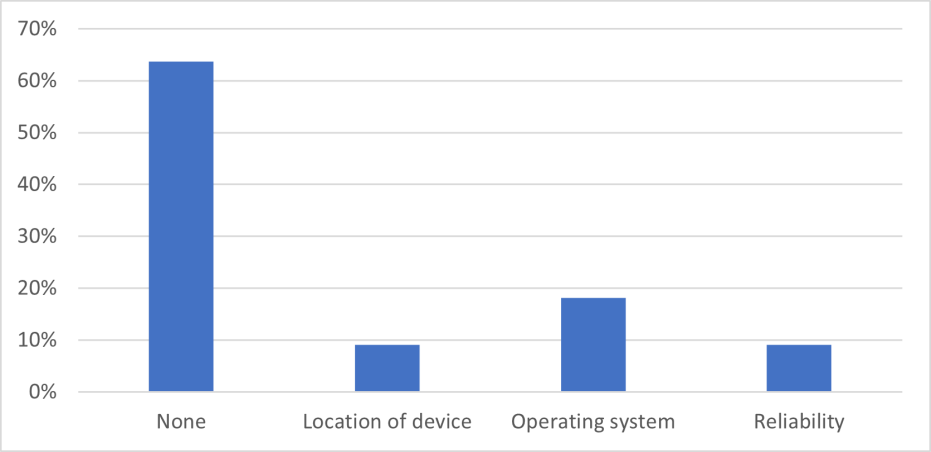

Fig. 4.10 Suggested Requirements for Edge Computing Layer

Then the participants were asked about their suggestions in terms of consumer requirements for Cloud computing. The results show that (82%) of the respondents had no suggestions, (9%) of them suggested adding the failure rate

and probability, and (9%) suggested adding the location of the Cloud datacentre. The majority were satisfied with the predefined list of vocabulary to express consumer requirements for Edge computing (See Figure 4.11).

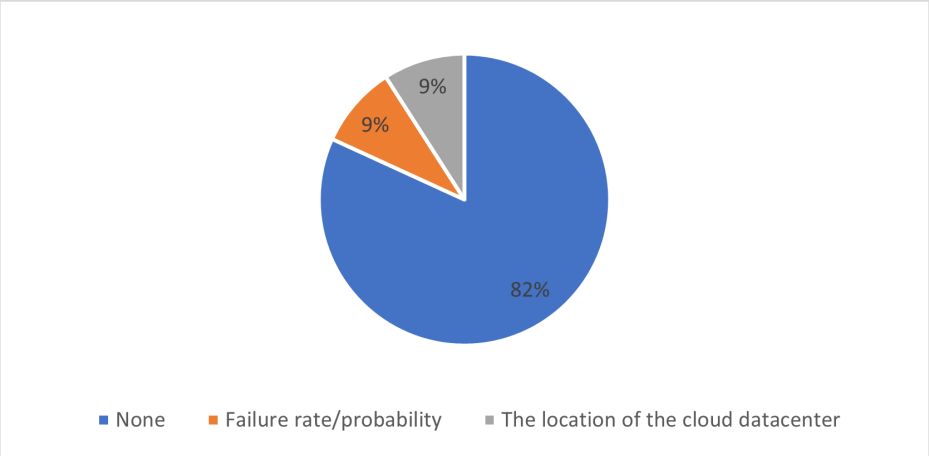

Fig. 4.11 Suggested Requirements for Cloud Computing Layer

Regarding consumer requirements for the sensing service, the results show that the list of requirements satisfied (45%) of the respondents. However, there were some additional requirements such as data quality (18%), data reliability (18%) and hardware and network capability (18%). This means that the list of consumer requirements for sensing services needs to be expanded (See Figure 4.12).

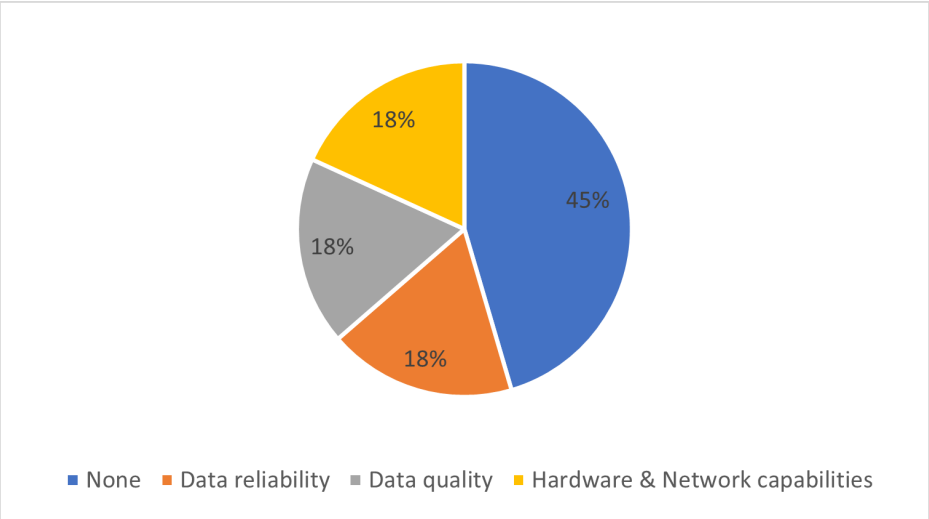

Fig. 4.12 Suggested Requirements for Sensing Service

The next question was about the consumer requirements of networking services. As indicated in the figure below (Figure 4.13), the majority (82%) were satisfied with the current list, some suggested additions related to networking configuration (9%) and data confidentiality (9%). This means that the current list of consumer requirements of networking services is sufficient.

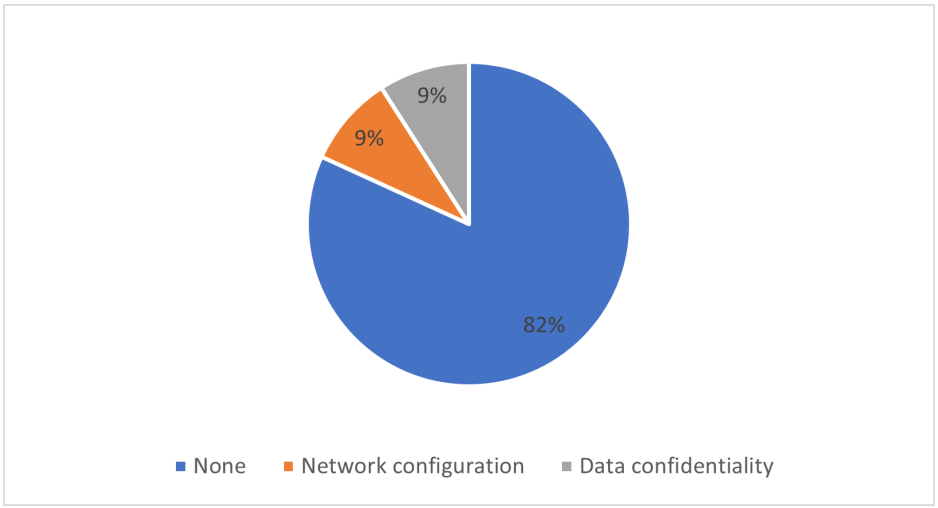

Fig. 4.13 Suggested Requirements for Networking Service

The next question was about consumer requirements for ingestion services. All of the respondents (100%) were totally satisfied with the current list and believed that nothing needed to be added.

Then the participants were asked about their suggestions regarding consumer requirements for machine learning services. As indicated in the chart below (Figure 4.14), just more than half of the respondents (55%) had nothing to add, while others suggested adding confusion matrix (9%), training (18%), feature extractor (9%), and efficiency (9%). As a result, the list of consumer requirements for machine learning services needs some additions.

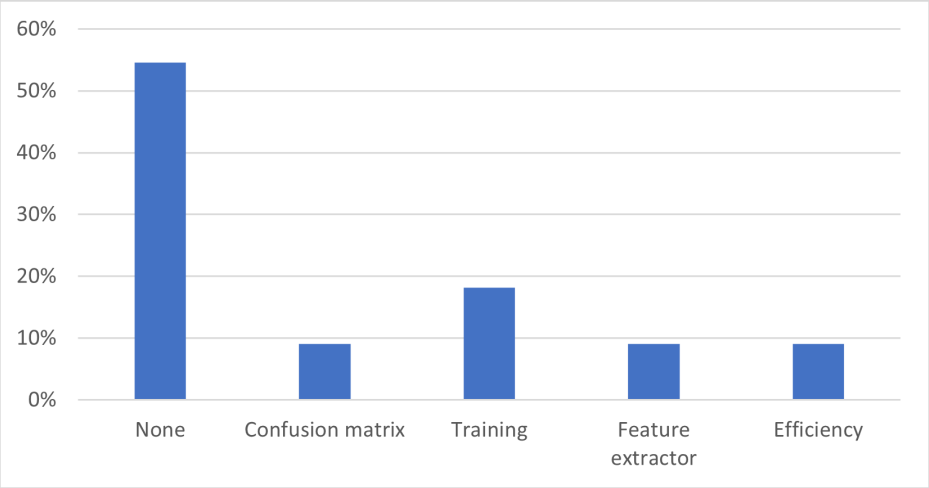

Fig. 4.14 Suggested Requirements for Machine Learning Service

The next question was about the list of consumer requirements for stream-processing services. As indicated in the figure below (Figure 4.15), (82%) of the participants had no suggestions, (9%) suggested adding publish/subscribe support, and (9%) suggested a data reduction factor/ratio. The majority were quite satisfied with the list of consumer requirements for stream-processing services.

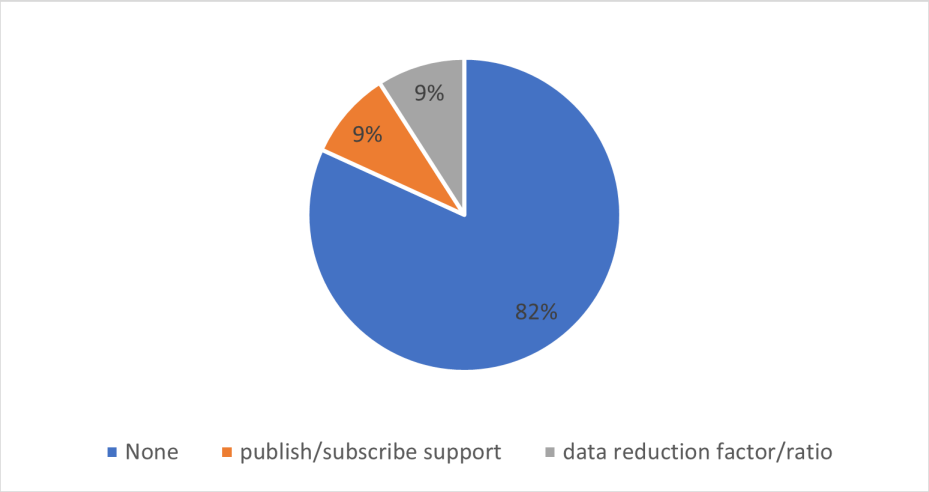

Fig. 4.15 Suggested Requirements for Stream-processing Service

After that, the participants were asked to add their suggestions regarding consumer requirements for batch-processing services. The majority of the respondents (82%) made no suggestions for additions, while others suggested batch running frequency (9%), and reducing and mapping failures (9%) (see

figure below). The list of consumer requirements for batch-processing services is therefore quite sufficient (see Figure 4.16).

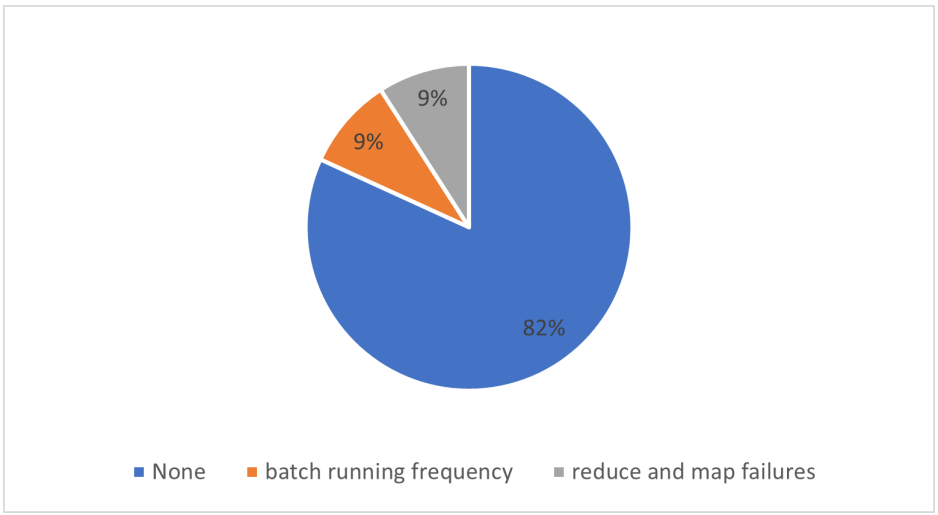

Fig. 4.16 Suggested Requirements for Batch-processing Service

Then, the participants were asked to add their suggestions regarding consumer requirements for SQL database services. The majority of the respondents (82%) had nothing to add, but some suggestions were made regarding data indexing support (9%), and version (9%). The list of consumer requirements for SQL database services is quite sufficient.

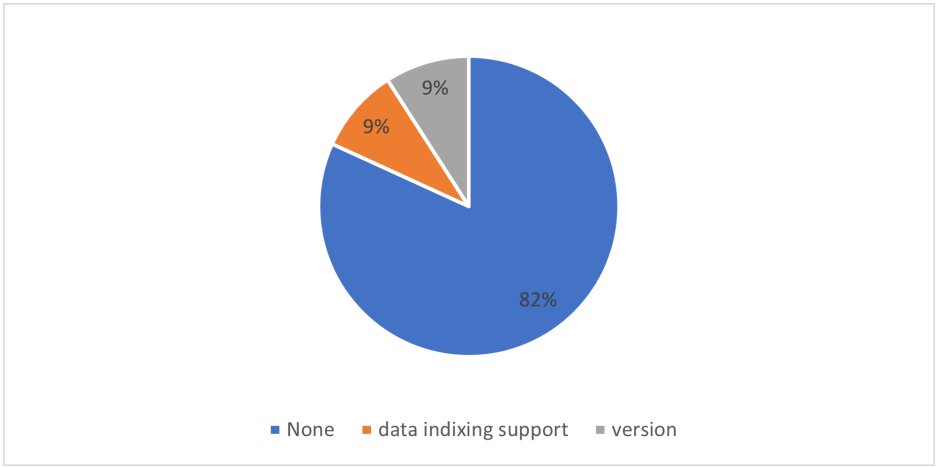

Fig. 4.17 Suggested Requirements for Database Service

The final question was about consumer requirements for NoSQL database services. All of the respondents (100%) were totally satisfied with the current list and believed that nothing needed to be added.

The participants tended to show dissatisfaction between 20%-50% for many questions, Figure 4.18.

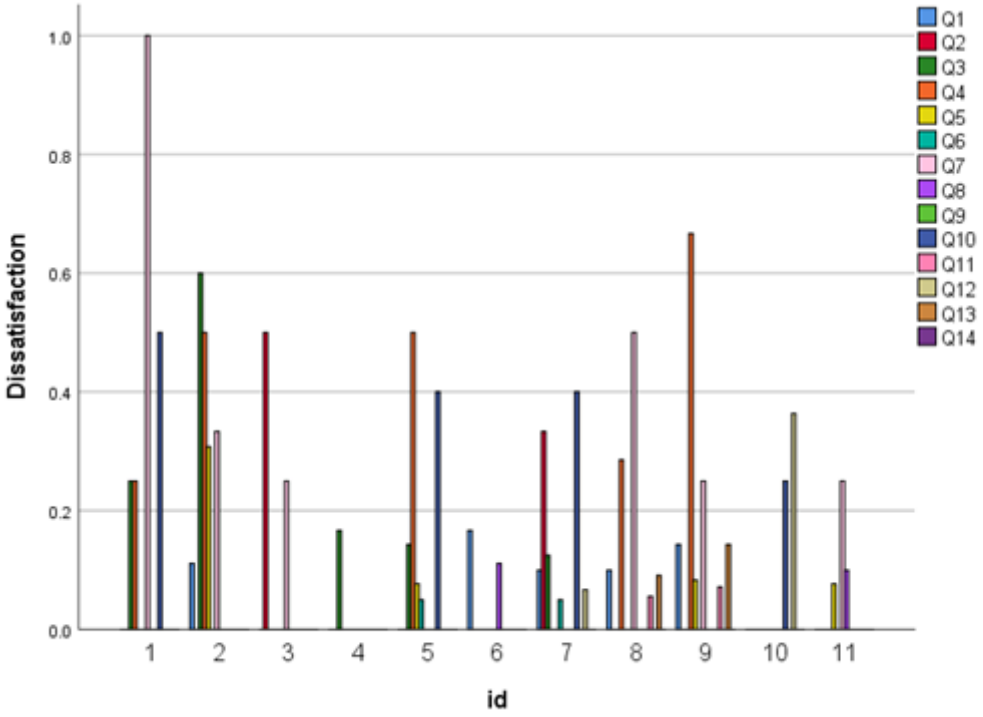

Fig. 4.18 Distribution of Dissatisfaction in 14 questions among 11 participants

Here, focused on the participants who mentioned missing/suggested activities, Figure 4.19 and Table 4.10 (Note: N represent number of participants who have suggested adding more requirements and SD represents Standard Deviation[6]). 45.45% of the participants mentioned missing/suggested activities in Q1, with dissatisfaction = 12.41%. 18% of the participants and 45.45% of the participants with dissatisfaction =41.66% and 25.69% for Q2 and Q3 respectively. Overall dissatisfaction of Goal1 reached 33.67%. For Goal two, missing/suggested requirements were obviously assigned to Q4 (45.45% of the participants), Q5(36.36% of the participants) Q7 (54.55% of the participants), and Q10 (36.36% of the participants) with dissatisfaction average =44.04%, 13.62%, 43.05%, and 31%, respectively. Overall dissatisfaction of Goal2 reached 19.95%. Based on the reflected percentage of the overall dissatisfaction for both Goal1 and Goal2, there is a high level of satisfaction for the presented list of requirements for both

[6]Standard deviation is a statistic that calculates a data set's variability relative to its mean and is measured as the square root of the variance

goals.

The suggested requirements by participants are taken into consideration, by refining the listed vocabularies as well as adding the suggested ones when it is possible.

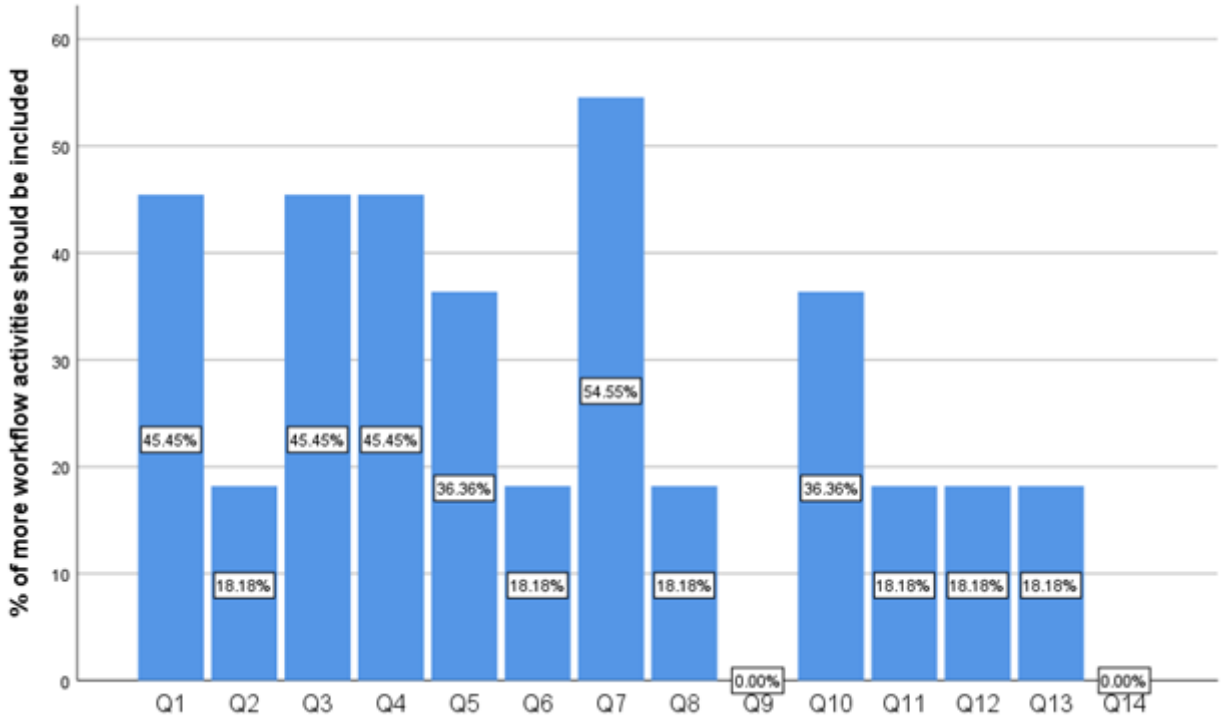

Fig. 4.19 Distribution of participants who mentioned missing/suggested vocabularies in 14 questions

Table 4.10 Descriptive statistics for 14 questions for participants who mentioned missing/suggested requirements

Question	Metric of question		
	Mean	N	SD
Q1: workflow activities	.1241	5	.0295
Q2: computing layers	.4166	2	.1178
Q3: services	.2569	5	.1976
Overall dissatisfaction of Goal1	0.3367		
Q4: IoT devices	.4404	5	.1721
Q5: Edge Computing	.1362	4	.1143
Q6: Cloud Computing	.0500	2	.0000
Q7: sensing services	.4305	6	.2954
Q8: networking services	.1055	2	.0078
Q9: ingestion services	-	–	
Q10: machine learning services	.3100	5	.1949
Q11: stream processing services	.0634	2	.0112
Q12: batch processing services	.1434	3	.1935
Q13: SQL database services	.1168	2	.0367
Q14: NoSQL database services	-	-	-
Overall dissatisfaction of Goal2	0.1995		

4.4 Comparison with Other SLA Languages

Due to the limited research efforts related to an SLA specification language specifically for the IoT, we compared our proposed language with the most commonly available service contract languages of Cloud and web services mentioned above (in the Introduction section). We used the following main criteria. The selected criteria were related to the scope of our solution and were not in any way conclusive. Our results are presented in Table 4.11:

- IoT domain: This criterion defines whether or not a language has been developed for the IoT domain.
- Syntax: This criterion is supported when there is a formal definition of the syntax, e.g., using BNF.
- Expressiveness: This criterion can be said to be met when the language contains a domain-specific vocabulary. If it does not provide a domain-specific vocabulary, then the expressiveness criterion is `partially` supported.
- Ease of use: This criterion can be viewed from the perspectives of developers and service consumers. From the service consumer perspective, ease of use is achieved if the user is not required to have much knowledge about how to create the specification in a machine-readable format. From the developers' perspective, ease of use is determined by whether the specification is written in a machine-readable format. The ease-of-use criterion is only `partially` met if just one of these perspectives has been considered.
- Supports different types of computational resources: This criterion is fully supported when a language considers the specification requirements of a range of resources, such as IoT devices, Edge resources, and Cloud resources, and `partially` supported when it allows only one category of required resources to be specified, such as only VMs.

Although many SLA specification languages for various application domains exist, we believe that in their current formats, they cannot accommodate the unique characteristics of the Cloud-based IoT domain. As can be observed in our comparison shown in Table 4.11, none of the compared SLA languages provide support for IoT applications. We attempted in our specification to consider the most

common/typical IoT application layers, including data sources, the most common data analysis programming models, and computational resources (e.g., IoT, Edge resources, and Cloud resources). Furthermore, there are different application models that have different stacks of essential interdependent services. For example, some applications require a certain type of data analysis programming model, such as applying data ingestion and stream processing to monitor a patient's health remotely. Other applications compute the statistics of a particular vehicle for a month-long period and require ingestion, stream processing, and batch processing data analysis programming models. Therefore, our SLA logic follows the workflow of IoT-based applications to simplify the process for users (e.g., IoT administrators) to specify their requirements. Our SLA logic enables users to select the workflow of activities for their IoT-based applications as well as to specify their requirements for each service and its computational/storage resources (e.g., the latency limit of the stream-processing service and the number of VMs). We developed a GUI-based tool (presented in the next chapter) to enable consumers to specify their requirements. The tool creates the SLA in a JSON format. By providing a GUI, we ensure the correctness of the SLA specification syntax. Most previous efforts provide the SLA template in XML format without the support of a GUI, which makes the process of creating a detailed and accurate SLA difficult. Furthermore, XML is not lightweight language.

Table 4.11 Comparison of SLA languages. Black circles represent features supported in the language, empty circles represent a partially supported feature and a hyphen (-) means not covered [27]

Comparison Features	WSLA	WS-Agreement	SLA*	SLAng	XCLang	CSLA	SLAC	iAgree	SLA-IoT
IoT Domain	-	-	-	-	-	-	-	-	●
Syntax	●	●	●	●	●	●	○	●	●
Expressiveness	○	○	○	●	●	●	●	○	●
Ease of use	○	○	○	○	○	○	○	●	●
Support different type of computational resources	○	○	○	○	○	○	○	○	●

4.5 Conclusion and Future Work

Defining “SLA offers” and “SLA requests” using standard vocabularies eases the process of comparing the available options and selecting the most suitable SLA offer based on consumer requirements. Therefore, we formally propose a syntax grammar to define an SLA specification language for the IoT paradigm. It is based on the proposed conceptual model (see Chapter 3).

Furthermore, we carried out a user study with domain experts to evaluate our proposed SLA specification. We applied the GQM approach to assess the significance of the vocabularies identified for an SLA specification of an IoT ecosystem. The participants were researchers whose research interests were focused on Cloud computing, networking and the IoT. The results demonstrate a high degree of satisfaction with the generalizability and expressiveness of the considered domain-specific vocabularies.

There are some limitations, such as the number of participants, that we will try to expand on in the future. Furthermore, there is the possibility of adding more vocabularies to capture QoS constraints and configuration requirements as well as considering more workflow activities. However, we developed a tool (presented in the next chapter) that allows us to extend vocabularies, workflow activities and computing layers. In future work, we aim to adopt a semantic-based approach to SLAs for different purposes by building an ontology for SLA specification for the IoT, derived from the proposed work. The ontology can express concepts of knowledge and the relationships between them, which can then be used for semantic analysis and verification purposes.

Chapter 5

SLA Specification Tool for IoT Applications

Overview

In this chapter, we demonstrate a toolkit for creating SLA specifications for IoT applications. The toolkit is used to simplify the process of capturing the requirements of IoT applications. We demonstrate the toolkit using the RHMS use-case and then evaluate the toolkit following a goal-oriented approach.

5.1 Introduction

There are a number of different approaches for specifying an SLA, ranging from employing a natural language or a formal language for the purpose of analysing SLA properties to utilising XML documents in an effort to standardise SLAs to increase SLA interoperability between the service consumer and the service provider [166]. For example, Keller and Ludwig provide an XML framework to express SLAs for web services (WSLAs), which is considered to be a starting point, as others have extended their approach [228]. Furthermore, some efforts in SLA specifications have been made for the Cloud computing paradigm, such as in CSLA [245] and SLAC [476]. However, SLAC [379] considers only IaaS, while CSLA [245] consider all three Cloud delivery models (IaaS, PaaS, SaaS).

These proposed works, as far as we know, allow users to type their SLA or use a predefined template and edit it, but none of them provide a GUI-based

tool that consumers can use as a wizard to create their SLA clauses. Therefore, in this chapter, we aim to present a toolkit that allows consumers (e.g., IoT administrators) to specify their SLA using GUI features. The tool considers the most common or typical IoT application tiers and services, as captured within the proposed grammar (presented in the previous chapter) in such a way that interested users can specify their preferences.

The tool is used to simplify the process of capturing the requirements of IoT applications and it supports the following: (1) specifying the service-level objectives (SLOs) of an IoT application at the application level; (2) specifying the workflow activities of the IoT application; (3) mapping each activity to the required services and infrastructure resources and specifying the constraints of the SLOs and other configuration-related metrics of the required services and infrastructure resources; and (4) creating the composed SLA in JSON format.

In the following, we present the design goals and the architecture of the tool. *The main contributions of the toolkit are as follows:*

- New vocabularies: The tool is based on a predefined grammar that consists of new vocabularies to express the SLA of an IoT application for common services (ingestion, stream processing, batch processing) as well as the infrastructure resources (IoT, Edge, Cloud). Providing new vocabularies allows for fine-grained SLA specifications for IoT applications, especially because, to the best of our knowledge, this is the first work to consider different computation layers (IoT, Edge, Cloud) within the SLA specification language for IoT applications.
- A GUI-based SLA specification tool: The SLA specification tool aims to provide a GUI that enables consumers to specify their requirements and then create the SLA in a machine-readable format (JSON format). By providing a GUI-based specification tool, the correct syntax can be ensured for the SLA specification, to some extent. Furthermore, the tool relieves users of the burden of specifying requirements in a machine-readable format such as JSON or XML, since it creates the SLA in a JSON format based on the user's specifications using the GUI (Most previous studies provide an SLA template in XML format [166][476][245][228]).

- Utilise Microsoft Excel to provide extendibility features for the tool: Workflow activity, the attributes related to SLOs and configuration metrics as well as the value ranges of the attributes are stored using Microsoft Excel. One of the reasons behind utilising Microsoft Excel is to make the tool extendable. Users do not need to change the Excel file, but if developers want to extend the tool, they have the ability to change/update/delete Rows and/or Columns in the Excel file with no need to change the code. Another reason to utilise Microsoft Excel, in addition to enhancing the extendibility feature of the tool, is its popularity and familiarity among the majority of users. Thus, it is available on most devices, which reduces the burden of having to download a prerequisite for the tool. Furthermore, it is a GUI-based software where users do not have to be experts in any related programming language details.

In the following sections, we present the design goal of the tool in Section 5.2 followed by the system architecture, which is presented in Section 5.3. Section 5.4 presents an evaluation of the tool from the consumer perspective. We conclude the chapter in Section 5.5.

5.2 Design Goals

SLA creation is an important and critical step, considering the fact that SLA-based service discovery, negotiation, monitoring, management and resource allocation rely on what is specified within the SLA. As a result, we have developed a toolkit that enables service consumers/providers to specify their QoS requirements and express them as SLOs, as well as specifying some configuration-related metrics for each service and infrastructure resource of the system. We consider the following features to be the design goals of the tool:

- Expressiveness: We aim to provide a rich list of domain-specific vocabularies to allow fine-grained SLA specifications.
- Generalizability: We aim to consider common components or layers of IoT architecture (IoT, Edge and Cloud).
- Extendibility: We aim to make the tool extendable to some extent by designing it in a way that allows anyone who is interested to customise/enhance

the SLA according to his/her application-specific needs to add or delete activities/metrics without changing the programming code. It is possible to add/delete/change activities/metrics using an attached Excel file, and these changes can be reflected dynamically. The Excel file preserves the schema of the SLA (e.g., workflow activities and their defined attributes).

- Simplicity: Providing a GUI enables users to specify their requirements without prior knowledge of how to write correct syntax for a machine-readable language such as JSON or XML. Furthermore, the tool allows users to specify an SLA in the same data flow as their application by specifying the workflow activities of their application first and then specifying the requirements in the same flow of occurrences as the selected activities.

5.3 System Architecture

The abstracted design and architecture of the tool are depicted in Figure 5.1. The overall architecture comprises three basic layers:

- The GUI layer, which includes the user interface components. The GUI layer displays a sequence of forms that guide the user through well-defined steps.
- The programming layer, which encapsulates the programming modules to serve the GUI layer by providing the required functionalities.
- The data layer, which encapsulates the required data as an input to the tool or an output of the tool. It includes the following:
  - An Excel file as an input, which provides data that describe the SLO and the configuration metrics related to the services and infrastructure resources of each activity.
  - A JSON file as an output, which represents the SLA specification.

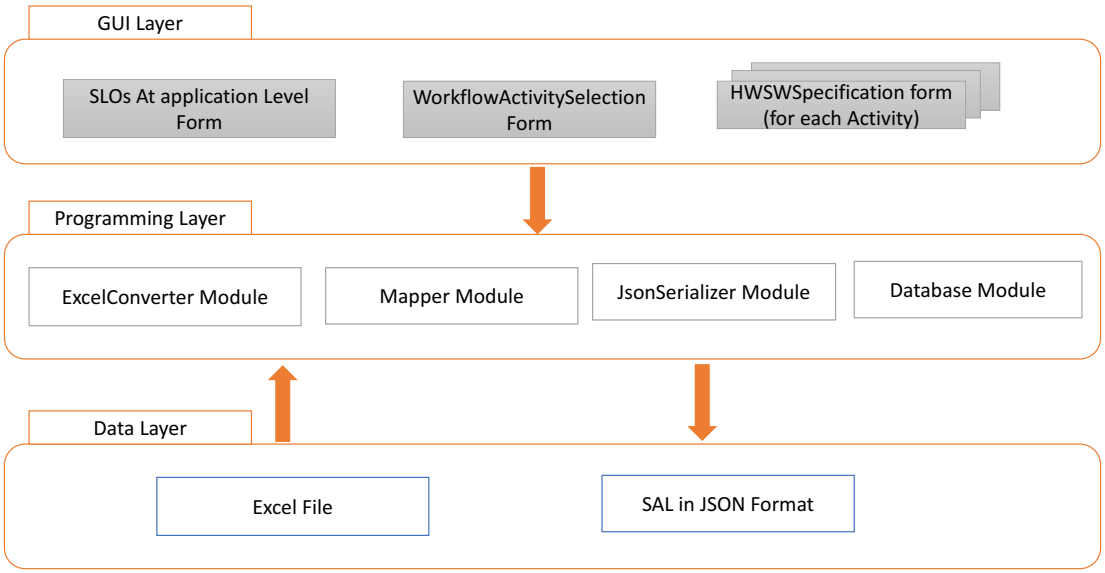

Fig. 5.1 The layered architecture of the tool

Figure 5.2 shows the sequence diagram of the tool and reflects the main steps when generating an SLA.

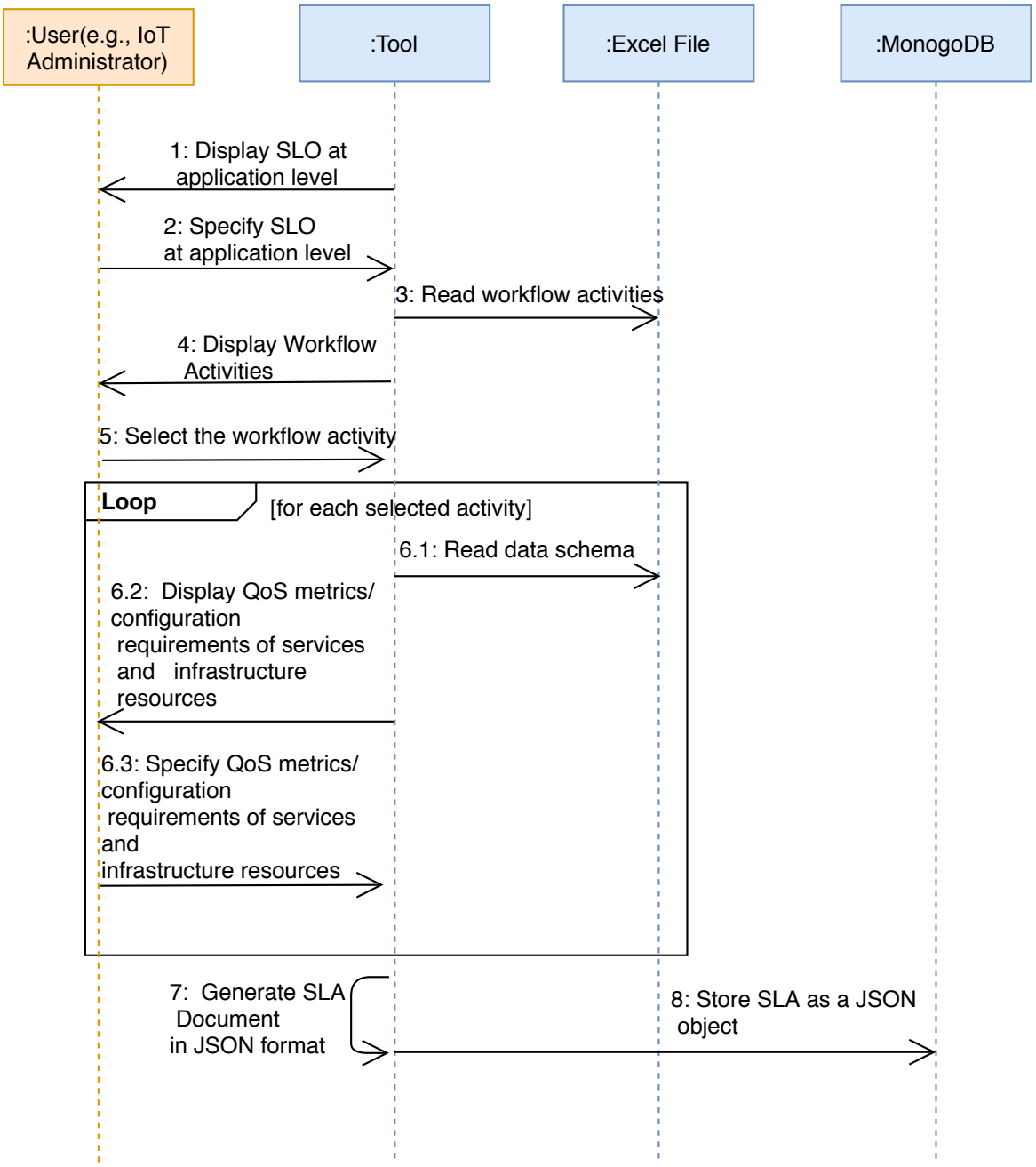

Fig. 5.2 Sequence diagram of the tool

The following section describes the implementation logic of the tool by listing the main steps that the user may experience when using the tool to generate the SLA:

- Step 1 -*Specify the SLO at the application level:* The tool displays a predefined list of possible SLOs as a checklist, and users can check the SLOs that they are interested in and specify the priority level (high, low, or medium), as well as the threshold value of the QoS metric of the SLO (see Figure 5.3).

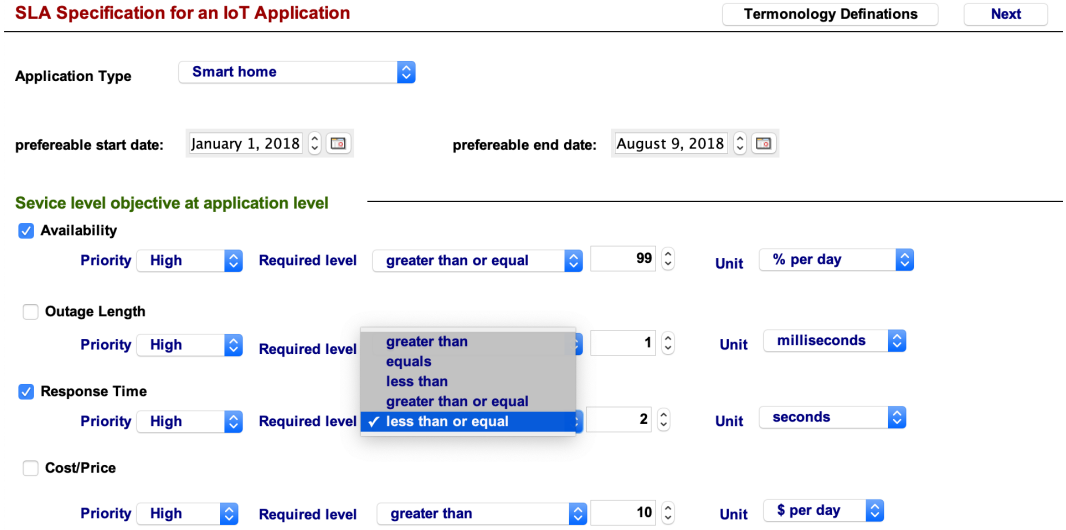

Fig. 5.3 Step 1: Specify Service-level Objectives at the application level

- Step 2 -*Select the workflow activity based on the application scenario requirement:* There is a predefined list of activities that are part of the workflow of a considerable number of IoT applications (e.g., capture event of interest; ingest data; analyse large-scale real-time data activity). The tool displays the predefined activities, and the user can then select the ones that are included within his/her application workflow activities and connect them in a way that reflects the data flow of the application. Connecting the activity preserves the dependencies between activities for future work related to performance modelling (see Figure 5.4).

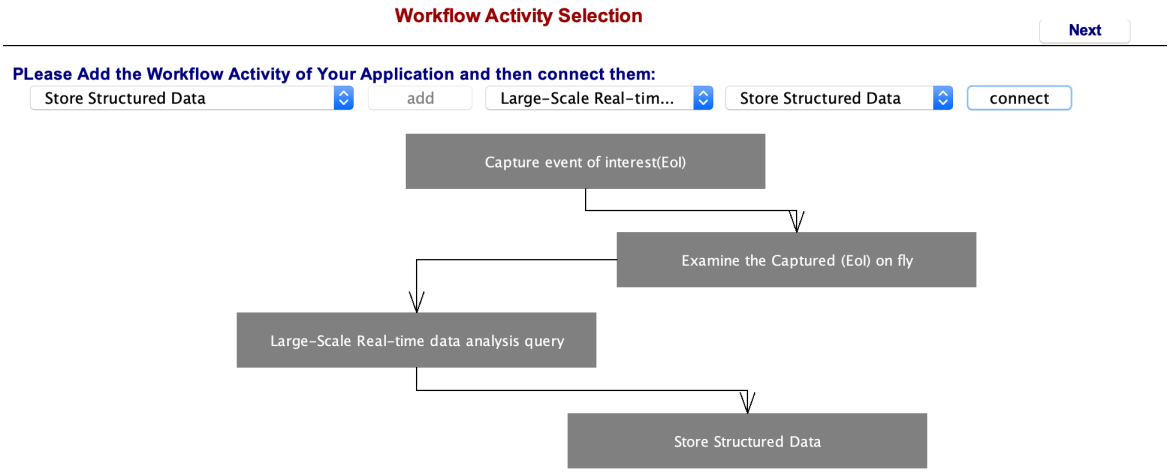

Fig. 5.4 Step 2: Select and connect the application workflow activities step

- Step 3 -*Map each selected workflow activity to its required service and infrastructure resource:* After selecting and connecting the workflow activities, the user can then specify, for each selected activity, the service and the infrastructure resource that host the service. For example, the “Capture Event of Interest” activity requires a sensing service that can be deployed on an IoT device (see Figure 5.5).

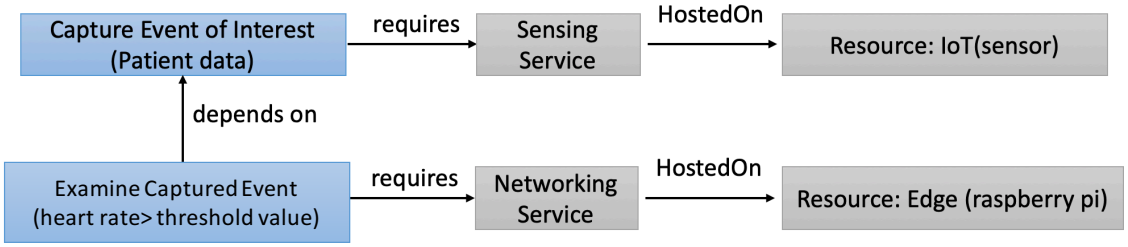

Fig. 5.5 Step 3: Map each selected workflow activity to its required service and infrastructure resource step

- Step 4 -*Specify the SLO and configuration metrics related to each of the selected activities:* The tool reads the SLO and configuration metrics schema from a predefined Excel file, which contains the schema content for each activity. The content of the Excel file also determines how the data are displayed dynamically. Then, users can specify the required level/value of an SLO and the configuration metrics for each service/resource infrastructure

component that is required to deliver the selected activities (see Figure 5.6).

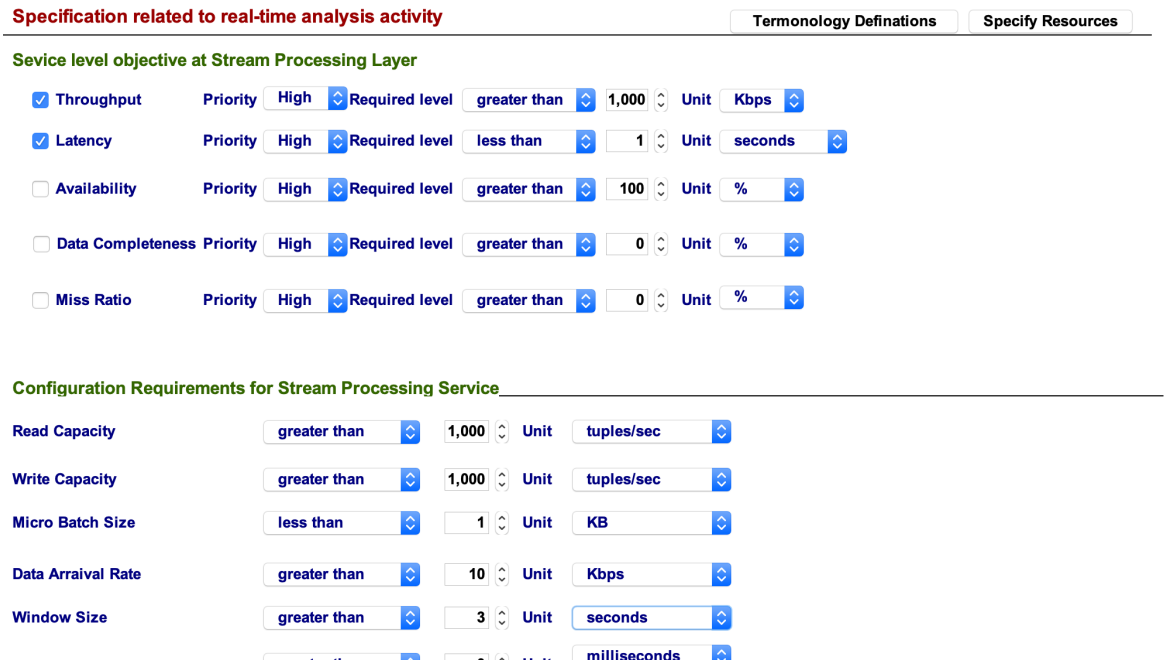

Fig. 5.6 Step 4: Specify the requirements of each selected activity step

- Step 5 -*Generate the SLA document:* Based on the user's specifications in the previous steps, the SLA will be generated in a JSON format (See Figure 5.7).

```
1  {
2   "appType" : "Smart health",
3   "startDate" : "Sun Oct 21 23:05:01 BST 2018",
4   "endDate" : "Mon Oct 21 23:05:01 BST 2019",
5   "slo" : [ {
6     "qosMetric" : "Availability",
7     "priority" : "High",
8     "requiredLevel" : "greater than",
9     "value" : "99.0",
10    "unit" : "% per day"
11  } ],
12  "slaid" : "Smart health Sun Oct 21 23:05:01 BST 2018Mon Oct
      21 23:05:01 BST 2019h"
13 }
```

Fig. 5.7 Step 5: Generate the SLA in the JSON format based on previous specifications

Figure 5.7 provides the basic details for the SLA of the RHMS, covering the application type and the start and end dates of the agreement. In addition, at line 5, it lists an SLO constraint at the application level; the snippet shows only the `Availability` as a high-priority objective with a required level greater than 99% per day.

- Step 6 -*Store generated SLA using a NoSQL database*: Users have the ability to choose which metrics to specify by checking/unchecking the metrics, which will result in different schemas for the JSON files being created due to the heterogeneity of requirements between users. Therefore, each generated SLA JSON file is stored in a NoSQL database (MonogDB) [1].

The terms used within the JSON are the same terms as those found within the grammar. The snippet covers some of the terms used within the first line in Table 4.1. Some other terms listed as production rules for the non-terminal $< SLA >$ (such as $< description >$) have not been used because they are optional.

The tool simplifies the process by guiding the user through the steps required to generate an end-to-end SLA and it can also be used to specify the requirements of different IoT applications. For example, the IoT administrator of the RHMS

[1] https://www.mongodb.com/

can specify the SLOs of the application, for example, that the response time for urgent cases should be less than 5 minutes. He/she will also be able to specify the activities involved, such as capture event of interest (e.g., patients' data), examine the captured events (for filtering), analyse real-time data on the fly and store the results that are of interest.

Figure 5.8 shows the process of mapping activities to the required service as well as the infrastructure resource for each of the involved activities. It also depicts an example of an SLO related to each of the required services and the infrastructure resources that are cooperating to deliver the RHMS.

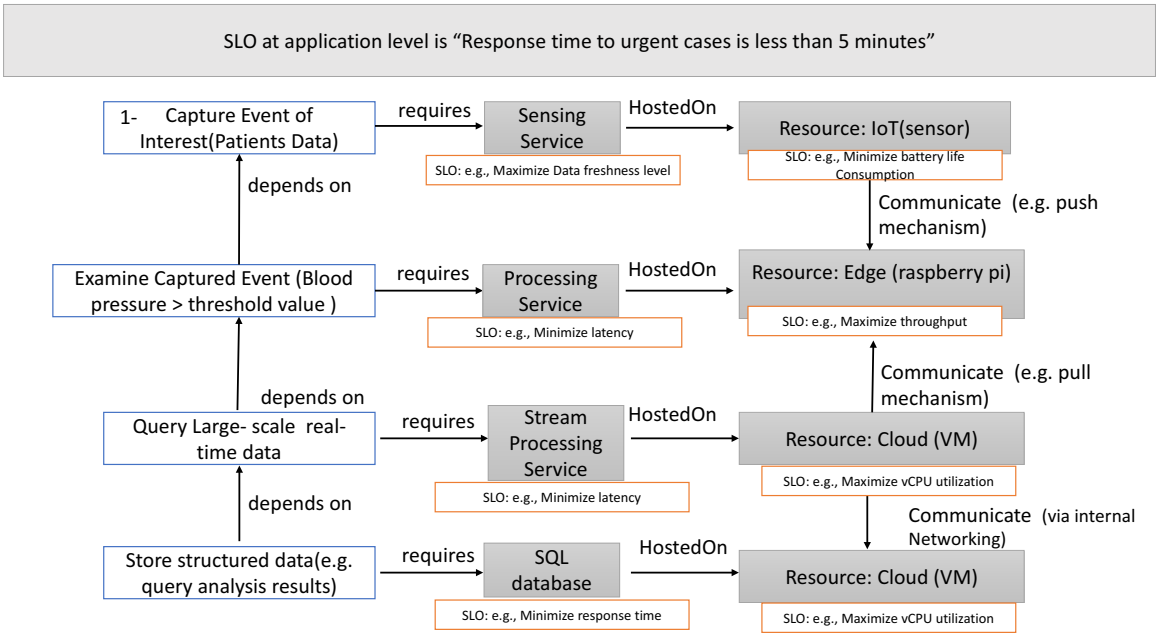

Fig. 5.8 Mapping activities to the required service as well as the infrastructure resource

Figure 5.9 shows the abstract structure of the main concepts that are considered within the resulting SLA document, with an example of each concept for clarification purposes.

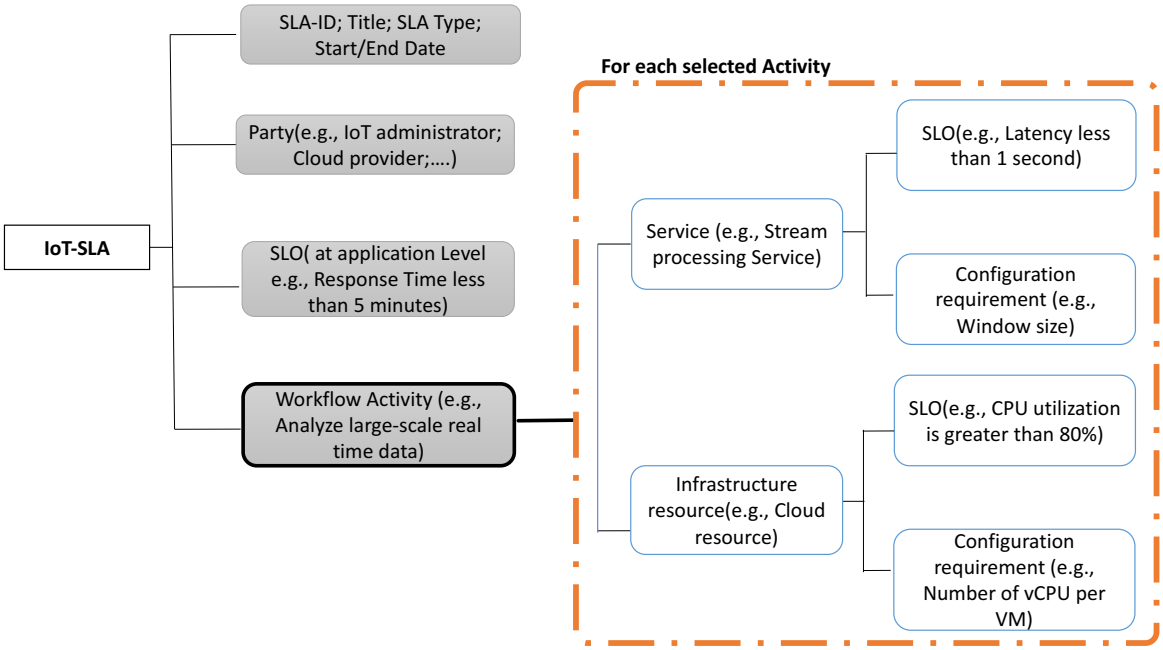

Fig. 5.9 The abstract structure of the main concepts that are considered within the resulting SLA document

5.4 Evaluation

We have evaluated our tool by testing to what extent we achieved the design goals that we set before we began developing the tool. We aimed mainly to develop a tool that meets the following goals: expressiveness, which allows for a fine-grained SLA specification; generalizability, which allows the tool to specify the requirements of different use cases; simplicity in reducing the effort needed from the end user to check the correctness of the syntax; and the extendibility of the tool.

The study was conducted with 14 participants, most of whom were PhD students working on topics related to the IoT, such as remote health and smart city applications. Their research interests included Cloud computing, Edge computing and networking. To allow the participants to reflect their opinions on the expressiveness and generalizability of the tool, we asked them to test a use-case that was related to their research. Moreover, we provided them with two use cases (RHMS and Flood Monitoring and Prediction System) in case any of the participants preferred to refer to a predefined use-case. They were also asked to

offer their thoughts on the ease of use of the tool.

We allowed the participants to try the tool. The output is an SLA in JSON format where they specified their constraints according to their research of interest use-case study. The participants were also able to discuss, ask for clarification and give instant suggestions if they had any. At the end, the participants were asked to fill out a paper-and-pen version of a questionnaire that contained four questions related to the tool. Furthermore, there was a comment text box to allow the participants to comment and offer their suggestions, criticisms or any other comments that they thought might improve the work.

The four questions related to the tool were as follows:

- Overall, how satisfied or dissatisfied are you with our tool?
- To what extent does the tool allow you to express your requirements? (i.e., measured based on the fine-grained level of expressiveness that the tool provides to specify the SLA constraints)
- How satisfied are you with the tool's ease of use?
- How satisfied are you with the tool's generalizability? (i.e., measured based on the level that you are able to specify the requirements of a use-case in your mind or use the use-case attached to the questionnaire)

5.4.1 Experiment results

The results of the participants' answers to the questions related to the tool are depicted in Figure 5.10. More than 60% described their overall satisfaction as "satisfied", more than 20% were "very satisfied" and less than 10% were "neither satisfied nor dissatisfied". Regarding the expressiveness (expressing user requirements) of the tool, more than 60% answered "mostly expressed", and the rest of the participants responded that their requirements were "fully expressed". From the ease-of-use perspective, 50% were "very satisfied", more than 20% were "satisfied", and more than 20% were "neither satisfied nor dissatisfied". Regarding the generalizability of the tool, 50% were "very satisfied", more than 40% were "satisfied" and the rest were "neither satisfied nor dissatisfied".

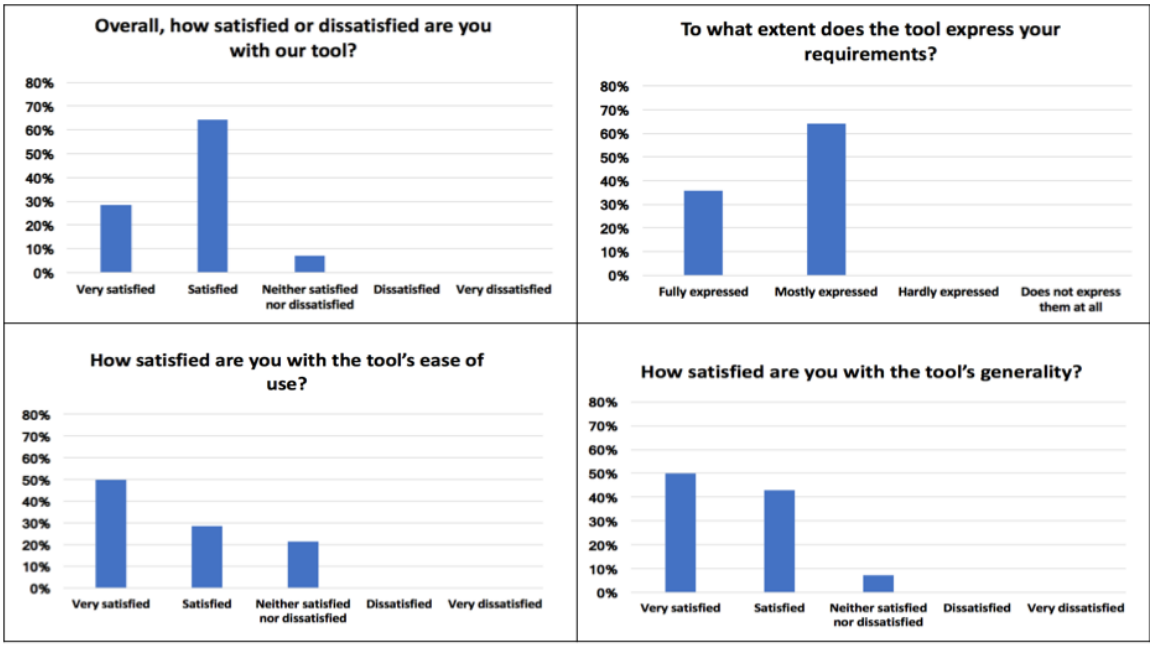

Fig. 5.10 Participants' responses to the questions related to the tool

5.4.2 Evaluation Analysis

Fourteen (14) participants took part in the experiment. As the answer for each question was based on a Likert scale, ordinal codes were assigned to the answers. For example, from very satisfied=1 to very dissatisfied=5, and from fully expressed=1 to does not express them at all=5. Percentages were used to explore the distribution of answers, while the median was computed to define the general tendency of the participants. The Wilcoxon test for one sample was used to compare the median to a hypothesised median value [132]. It was used, in this study, to examine whether there was significant satisfaction with the tool, its ease of use, and generality (median<=2), and the requirements of the conceptual model (median<=2). A p-value of 0.05 was used as the threshold for significant results.

The results in Table 5.1 indicate that the participants were very satisfied with the tool (median=1.50), which was very highly significant (p-value<.001). The participants found that the expressiveness (expressing user requirements) of the tool seemed to be fully expressed (median=1.50), which was very highly significant (p-value < 0.001). The participants were satisfied with the ease-of-use perspective (median $= 2.00$), which was very highly significant (p-value<.001).

There was satisfaction with the generality of the tool (median=2.00), which was very highly significant (p-value< 0.001).

Table 5.1 Result of conceptual model using the Wilcoxon test

	Median	p-value (Wilcoxon test)	Decision
Overall, how satisfied or dissatisfied are you with our tool?	2	<.001	Significant result
To what extent does the tool express your requirements?	2	<.001	Significant result
How satisfied are you with the tool's ease of use?	1.5	<.001	Significant result
How satisfied are you with the tool's generality?	1.5	<.001	Significant result

Since the participants were asked about their overall satisfaction and the generality of the conceptual model and the tool. Figure 5.11 presents a comparison between the overall satisfaction with the conceptual model and the tool. Figure 5.12 presents a comparison between the overall satisfaction with the generality of the conceptual model and the tool. As can be observed from the two figures, the participants were more satisfied with the tool, which might be explained by the fact that the conceptual model requires some understanding of UML notations while the tool is GUI based and thus provides more clarity than the conceptual model. However, based on the Wilcoxon test, the difference between overall satisfaction with the conceptual model and satisfaction with the tool was not statistically significant (p-value=0.102), and the same result was seen for overall satisfaction with the generality of the conceptual model and the tool (p-value=0.257),(Table 5.2).

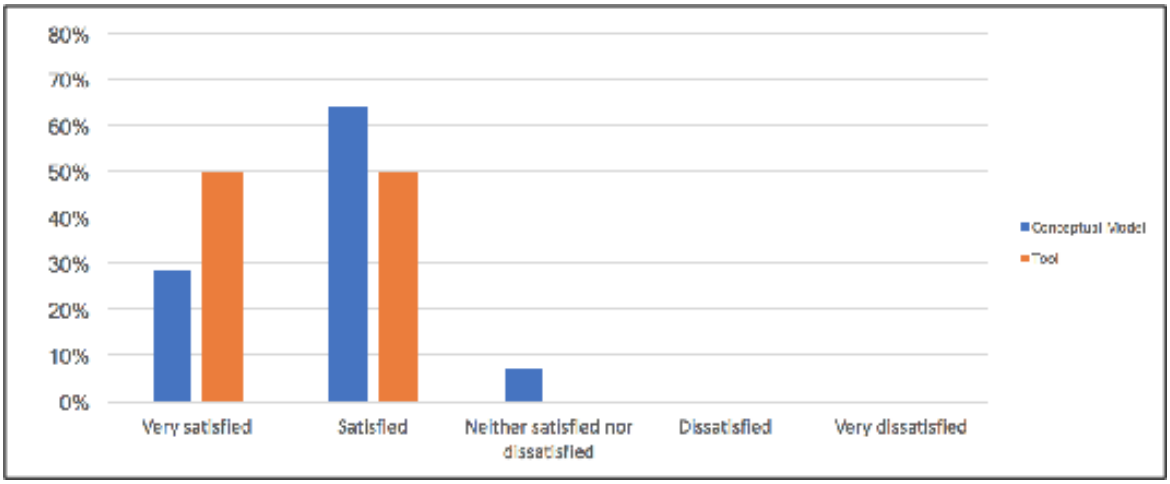

Fig. 5.11 Comparison between the overall satisfaction with the conceptual model and the tool

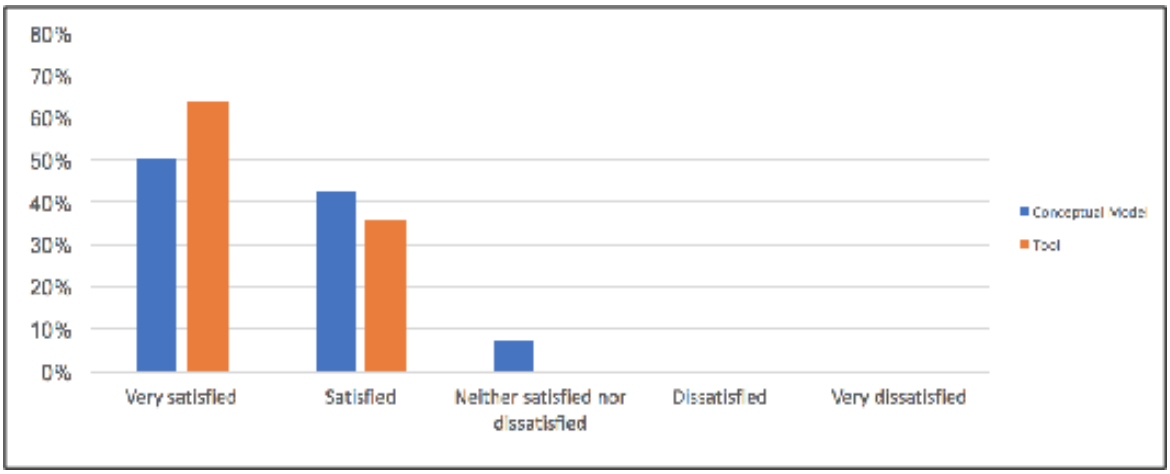

Fig. 5.12 Comparison that reflects how satisfied the participants were with the generality of the conceptual model and the tool

Table 5.2 Comparison of attitudes towards the conceptual model and the tool using the Wilcoxon test

Question	p-value (Wilcoxon test)
Overall, how satisfied or dissatisfied are you with our conceptual model? Vs Overall, how satisfied or dissatisfied are you with our tool?	0.102
How satisfied are you with the tool's generality? Vs Overall, how satisfied or dissatisfied are you with our tool?	0.257

There were a number of comments that are listed below:

- comment related to the listed vocabularies:
  - "It is better to add warranty duration as one of the parameters to be specified for IoT devices".
  - 'Window size' configuration metric can be expanded to cover different types of windows supported in stream processing, such as sliding Vs tumbling window, event-based Vs time-based window".
- Comment related to the design of the tool:
  - "Could have a hierarchy window, just to reflect which specifications are being done now".

Regarding extendibility, we tested it by reviewing the results of a case conducted by a master's degree student who was interested in integrating security constraints into the SLA clauses. The security metrics were integrated only by inserting the related vocabularies using the attached Excel file, with no need to change any line within the code.

However, regarding the question: "To what extent does the tool allow you to express your requirements? (i.e., measured based on the fine-grained level of expressiveness that the tool provides to specify the SLA constraints)", the answers could be partially affected by subjectivity. Nonetheless, the participants were asked to base their answer on whether or not the tool allowed them to

specify the requirements that they were aiming to consider within the use-case related to their research interest. Even with the possibility of subjectivity, we considered the result from the point of view of more than one participant and reflected the feedback of the majority. Thus, as Figure 5.10 shows, there is quite a high level of satisfaction with the expressiveness of the considered fine-grain domain-specific vocabularies.
Furthermore, based on the Wilcoxon test, regarding the expressiveness (expressing user requirements) of the tool, most of the participants found that their requirements are fully expressed (median=1.50). This was very highly significant (p-value < 0.001) (Table 5.1). We also compared this result with the previous chapter, where the participants reviewed the expressiveness of the considered vocabularies within the proposed grammar, and indicated a high level of satisfaction (see Figure 4.5). Thus, we believe there is a level of consistency in the evaluated expressiveness in the two separate studies.

5.5 Conclusion and Future Work

We have developed a tool that supports the end-to-end specification of QoS requirements within SLAs for IoT applications. The tool is used to simplify the process of capturing the requirements of IoT applications. We believe that the tool effectively tackles the aforementioned challenges: 1) The tool provides a rich set of vocabularies to capture the requirements of each layer of the IoT architecture (IoT device, Edge layer, Cloud layer) to overcome the heiteroginty. 2) IoT applications have different SLO requirements, which vary from one application to another. Additionally, the priority level of one SLO differs from one application to another. Therefore, the tool allows users to specify SLOs at the application level. 3) Different IoT applications have different workflow activities depending on each application use-case scenario. Therefore, to overcome the varied requirements that arise from the heterogeneity of workflow activity, the tool allows the users to select their workflow activities first and then specify the requirements for each selected activity. The output of the tool is an SLA specification in a machine-readable format (JSON format).

We have tested the tool with different use cases, and it allows requirements to be captured. However, there is a possibility that some use cases will not fit with this tool. In this case, we believe that the extendibility feature can mitigate this risk to some extent with no need to change any line of the code. The tool in this stage works for our purposes; however, as future work, we will enhance the tool to include more semantic-related conditions. For example, when specifying the end-to-end response time at the application level, the tool needs to consider that all specified response times for the involved activities should not exceed the acceptable end-to-end response time at the application level. Thus, there is a need to consider the inter-dependency between the activities. When activities are running in a sequential order, the total response time of all activities should not exceed the end-to-end response time. On the other hand, when activities are running in parallel, the activity with the highest response time among them should be less than the specified end-to-end response time. For further illustration, consider a case with a combination of parallel workflow activities (p1, p2, p3) and sequential workflow activities (s1, s2, s3). When an IoT administrator specifies an end-to-end response time of an IoT application that should be less than or equal to Y time units, then the tool should ensure that the following formula is maintained:

$$\sum_{i=0}^{i=3} ResponseTime_{s_i} + (\max_{1 \leq j \leq 3} ResponseTime_{p_j}) \leq Y \tag{5.1}$$

where $ResponseTime_{s_i}$ means the specified response time of workflow activity s_i where i=1,2,3 and they are activities that can be performed sequentially.

where $ResponseTime_{p_j}$ means the specified response time of workflow activity p_j where j=1,2,3 and they are activities that can be performed in parallel.

Chapter 6

Application Scenario Where the SLA Specification Tool Brings New Value for SLA Management

Overview

In this chapter, we propose an SLA management framework and present a proof of concept to reflect a number of SLA management phases in which the SLA specification plays a part. First, in the background section, we present two applications that utilize our SLA specification. In the first case, since the SLA specification provides configuration parameters for fine-grained details, it has been utilized to create the knowledge base of a context-aware recommendation system for the configuration of Cloud/Edge-based IoT applications. In the second case, a Java library has been implemented to translate the generated SLA from a JSON format to a smart contract. Then, these two applications are utilized as part of our proposed SLA management framework. The proposed SLA management framework consists mainly of the following phases: SLA specification, SLA negotiation, SLA monitoring using Blockchain to deploy SLA-based smart contracts, SLA enforcement and SLA compensation.

6.1 Introduction

With the spanning of services between IoT devices, the Edge layer and the Cloud layer, users need some guarantees that their QoS requirements, i.e., "a level of

quality that is agreed upon as a constraint" [289], will be assured and satisfied. Therefore, developing an SLA management framework is one of the solutions that can mitigate the risk of violating SLA terms.

SLA management includes more than one phase of the SLA lifecycle, such as SLA negotiation, monitoring and enforcement. There are a considerable number of studies in the literature covering SLA management. In Section 2.2, Table 2.2 provides a list of work mapped under the SLA management sub-category (48 references). However, all of these SLA management works are related to the Cloud. For example, Torkashvan et al [464] propose an SLA management framework for Cloud computing and inter-Cloud environments in particular. This framework is based on the WSLA implemented by IBM, but it has been altered to fit Cloud computing.

Zhao et al [525] present a new approach to the SLA-based management of Cloud-hosted databases. They present an end-to-end framework for managing Cloud-hosted databases. The framework promotes the adaptive and dynamic provision of the software application database based on application-defined constraints to meet the required SLA performance of the applications. The framework monitors the SLA continuously and, when required, automatically triggers the necessary corrective actions (database tier scaling out/in). However, their study considers SLA management only for database tiers. Mavrogeorgi et al [319] present a Cloud-based SLA management system. Their study provides an SLA enforcement mechanism based on rules, and these rules are updated in run-time to proactively detect and manage possible SLA violations.

However, there is a shortfall in the number of available feasible SLA management frameworks that develop a management mechanism for SLA lifecycle phases (e.g., definition, negotiation, monitoring and enforcement phases) for IoT applications. Furthermore, there is a shortfall in the use of Blockchain technology and combining the SLA specification with the recommender system, especially in terms of considering the complexity of the multi-layered nature of IoT applications.

In this chapter, we propose an SLA management framework that consists mainly of the SLA specification, SLA negotiation, SLA monitoring using Blockchain to deploy SLA-based smart contracts, SLA enforcement and SLA compensation. In addition, we add an optional configuration recommendation phase since, as explained in Chapter 3, some configuration parameters such as sample rates can affect the data analysis accuracy because they affect the data freshness. For this reason, defining the SLA parameters and their related configuration requirements requires expertise. As a result, having a recommender system that can be considered as a guide for most common configuration parameters for different infrastructure resources that span layers, provides considerable help for most contractual parties of IoT-based services.

In Section 6.2, we present two applications that utilize the SLA specification. In the first case, Section 6.2.2, since the SLA specification provides configuration parameters on fine-grained details, it has been utilized to create the knowledge base of a context-aware recommendation system for the configuration of Cloud-/Edge-based IoT applications. In the second case, presented in Section 6.2.3, a Java library translates the generated SLA from a JSON format to a smart contract that can then be deployed on a Blockchain network. Then, these two applications are utilized as part of our proposed SLA management framework, which is presented in Section 6.3. After that, Section 6.4 provides a proof of concept for the proposed SLA management framework. In Section 6.5, we discuss the proposed SLA management framework and compare it with other related studies. We conclude this chapter in Section 6.6.

6.2 Background

6.2.1 Hyperledger Fabric

Hyperledger Fabric is an open source application of a Blockchain framework. It contains a modular architecture that allows Blockchain developers/administrators to deploy a Blockchain network using multiple consensus protocols and membership options [210]. Hyperledger Fabric also contains a smart contract engine capable of performing Java, JavaScript and Go smart contracts. Hyperledger Fabric enables consortium blockchains to be developed and deployed

by offering a set of functionalities that enables an administrator to generate different certificates and settings files, which are then used to build a Blockchain network. In Fabric, administrators can select different database technologies to store data, such as 'LevelDB' and 'CouchDB', due to the plug-and-play architecture of Fabrics. Furthermore, the administrator can define membership policies as well as the consensus protocol that can be applied.

6.2.2 IoT-CANE (Context-Aware recommendatioN systEm)

Our proposed SLA specification allows the consumer to specify fine-grained configuration parameters and SLO constraints at the application level, as well as at each workflow activity level. However, consumers interested in deploying IoT services, for example small organizations that want to deploy IoT services and need to submit an SLA request to one or more service providers (network provider, Cloud provider), need to specify the required configuration parameters for the required services and/or resources along with the SLO constraints at the service and/or application level. The task of specifying fine-grained configuration parameters requires some experience in the IoT, the Cloud and networking. One of the novelties of the proposed SLA specification is that it considers the fine-grained details for most common configuration parameters of the most common workflow activities. Thus, there is potential usage within a recommender tool for the configuration parameters of an IoT application. Therefore, Li at al [274] develop the IoT-CANE (Context-Aware recommendatioN systEm), which utilizes our SLA specification tool to generate the knowledge base [1].

The IoT-CANE solution introduces an end-to-end pipeline, proposed in [274], for the classification, configuration and recommendation of suitable solutions in this most complicated ecosystem. For the first time, a ripple-down rules (RDR) method is used to recommend an IoT-based configuration with a single conclusion/classification environment. In its processing layer, the IoT-CANE uses our SLA specification tool to create the base knowledge of the recommender for different possible configurations, which are used for later selection depending on the consumer's requirements. The created knowledge is stored in a configuration knowledge database.

[1] IoT-CANE is developed by a PhD researcher at Newcastle University (first author in [274]

Figure 6.1 shows the basic concepts that are considered within the Entity Relationship (ER) diagram of the recommendation rules. The dotted square reflects the context knowledge that is captured using the output generated from the SLA specification tool.

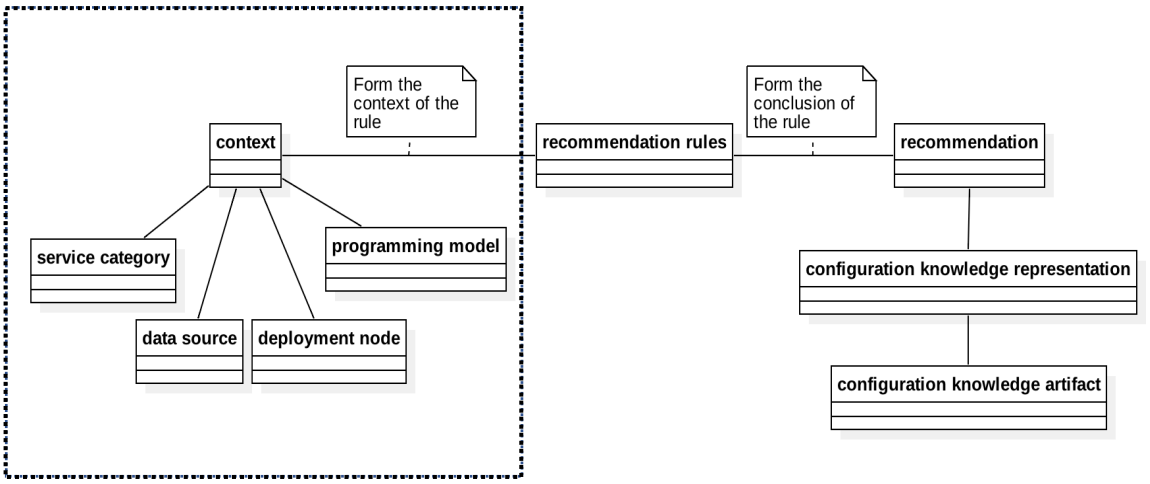

Fig. 6.1 ER diagram of recommendation rules [274]

The IoT-CANE maintains a configuration knowledge base for the IoT resources and it stores contextual data about the configuration knowledge representation. The recommendation rules, as shown in Figure 6.1, maintain an association between the items in the configuration knowledge base. The recommendation rules consist mainly of two parts: context and conclusion.

- Context: The left side of the ER diagram presents context information regarding the expected "contexts" data (e.g., sensing service), data source (e.g., sensors, social media APIs), programming model (e.g., streaming process, batch processing, SQL, NoSQL) and deployment node (e.g., Edge resource, Cloud resource). The purpose of the configuration knowledge base is to capture metadata and prevalent data on classes with similar demands for implementation and resources. Shared knowledge of the context allows IoT users to customize and reuse one of the previous configurations.

- Conclusion: The right side of the ER diagram depicts the parts that form the conclusion of the produced recommendation rules. The produced recommendation rules suggest configuration knowledge representation, which can then be deployed using a defined configuration deployment engine (e.g., Docker).

6.2.3 From SLA to Smart Contract Java Library

A Java library [2] converts the generated SLAs for different IoT application use cases with different SLO constraints to a smart contract (see Figure 6.2).

Generating an SLA in a machine-readable format simplifies the process of translating the SLA to Smart contracts. We aimed to translate the generated SLA into a smart contract to explore the use of Blockchain technology for monitoring IoT applications and recording SLA violations.

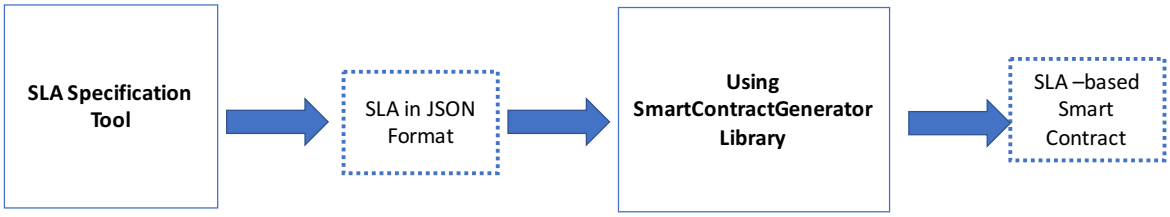

Fig. 6.2 Abstracted generated smart contract from SLA specification steps

Applying the FromSLAToSmartContract library to the SLA can create a list of rules for each SLO constraint and configuration requirement related to each workflow activity. There are two main classes in the "FromSLAToSmartContract" library:

- ChaincodeBuilder class: The rules express the constraints that have been specified within the SLA and they are used to validate the status of a monitored application. There is a ChaincodeBuilder class for each workflow activity and for the SLO parameters that reflect the QoS constraints at the application level. The ChaincodeBuilder extracts the SLO constraints and the configuration required for each workflow activity and then builds three chaincode methods: $*\_update$,$get\_latest\_*update$ and $get\_*\_violations$ For each of the SLO and configuration requirements:
  - $*\_update$: The method is used to reflect the current state of the examined SLO/requirement.
  - $get\_latest\_*update$: The method returns the most recent state reported.

[2] We refer to this library as "FromSLAToSmartContract" library . The FromSLAToSmartContract library converts the generated SLA from JSON format to a smart contract. It is developed by researchers at Newcastle University [85] and it is part of the first author's master's dissertation.

  - *get_∗_violations*: The method returns a history of all violations reported for a specific SLO/configuration requirement.

- ChaincodeGenerator class: The ChaincodeGenerator class serves as the API to the library. The ChaincodeGenerator class takes advantage of the chaincode builders described above. After all methods have been produced by the ChaincodeBuilder, the ChaincodeGenerator generates Java classes and appends the methods previously generated by the ChaincodeBuilder using JavaPoet's [3] features. Finally, the ChaincodeGenerator returns an object that contains the Java class as a string and the smart contract documentation. Furthermore, a deployable Hyperledger Fabric smart contract project is written to the file system, which can be deployed to the Blockchain.

6.3 Proposed SLA management Framework

In this section, we propose an SLA management framework. The framework is proposed as one possible solution to ensure that SLA violations are monitored, violations are recorded to demonstrate that our specification tool can play a part within more than one phase of the proposed framework. The proposed framework is depicted in Figure 6.3, and it consists mainly of the following phases:

[3] https://github.com/square/javapoet

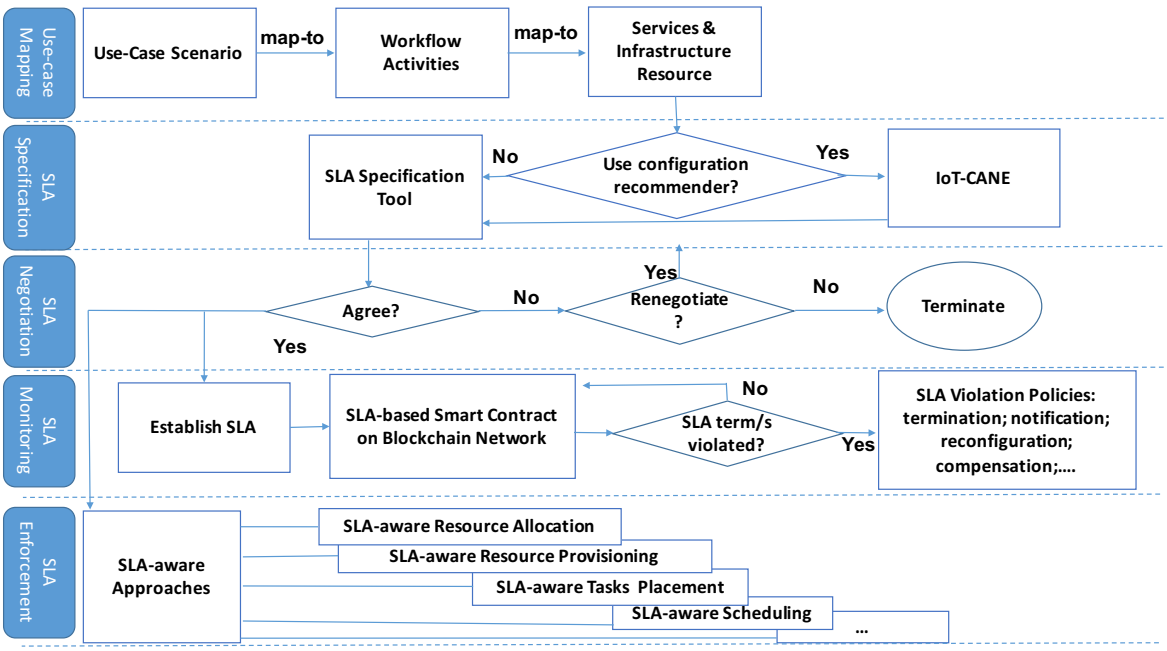

Fig. 6.3 Proposed SLA management framework

- Use-case mapping phase: This phase consists of the following steps:
  - Use-case Scenario: The scenario describes the type of application and the main cooperative activities/tasks that are required to deliver the final IoT application.
  - Workflow Activities: This step includes one or more of the activities that a use-case scenario can be mapped to. For example, in the flood monitoring and prediction (FMP) use case, some possible workflow activities are data collection, data examination and real-time data analysis.
  - Service and Infrastructure Resource: Each of the cooperative workflow activities can be associated with a required service and the infrastructure resource that hosts the service. For example, in FMP, the real-time analysis activity can be associated with a stream processing service.
- SLA Specification Phase: In the SLA specification, the service provider/s and consumer/s can specify the QoS constraints, price and actions. In the proposed SLA management, the SLA specification consists mainly of the following steps:

  - IoT-CANE: This step allows us to enter some context for each service that is associated with an activity and the infrastructure resource layer where the service will be hosted. We can then output the configuration parameters for the service and the infrastructure resource in a JSON format (a brief explanation of how the IoT-CANE works is presented in Section, 6.2.2). This optional step depends on whether the consumer (e.g., the IoT administrator) wants a second opinion about possible configuration parameters.
  - SLA Specification: This step allows us to specify agreement terms, considering SLO constraints as well as configuration requirements. However, the outcome of utilizing the IoT-CANE can be used to aid with specifying configuration requirements as a second opinion.

- SLA Negotiation: SLA issues, such as service quality, price or average response time, can be negotiated between a service provider and a consumer. There can be a gap between the consumer's expected demands (i.e., service level) and the level of service that can be provided by the provider. If this gap exists, the provider and consumer will then compromise to obtain a mutually agreed-upon level of service. The agreed-upon service level becomes part of the SLA when the negotiation is successful.

  In the proposed SLA management framework, SLA negotiation takes place before any required/offered services are established. If an agreement is reached, then the final agreed-upon terms can be established, and the SLA monitoring and enforcement phases start. If no agreement is reached, then if renegotiation is possible, the SLA specification phase is repeated; otherwise, the SLA renegotiation is terminated.

- SLA Monitoring: The expected service level between the consumer and the provider is included in the SLA contract. Nonetheless, the QoS criteria that are included in an SLA (such as response time and throughput) are vulnerable to change; thus, to enforce the agreement, these parameters must be monitored closely to determine whether the service provided meets the SLA QoS constraints.

Smart contracts and distributed ledgers could revolutionize the economy. Smart contracts, on the one hand, are agreements that are defined in an executable language to allow trusted transactions. A key feature of such contracts is that they can be implemented consistently by a network of distrusted nodes without the intervention of a third party [63, 474]. Smart contracts are commonly connected with distributed ledger technology, such as Ethereum, which is utilized for different areas, mainly finance but also Cloud services. On the other hand, the Blockchain [351, 474], which is an implementation of the distributed ledger concept, executes transactions and registers them in a decentralized manner.

Therefore, a potential usage of technology is to monitor the IoT using SLA-based smart contracts so that the monitoring process can be conducted transparently by all IoT application participants. This would mitigate the need for a third party to act as a monitoring service, which may in some circumstances be biased towards one of the interacting parties. Furthermore, through its consensus and security mechanisms, Blockchain provides a platform to ensure that agreed-upon SLA terms and any logged interactions are secure and unalterable.

Thus, in the proposed SLA management framework, for the SLA monitoring phase, the generated SLA is translated into a smart contract that can be deployed on a Blockchain network to automate SLA monitoring. This process has the following steps:

- Translate the agreed-upon SLA from a machine-readable SLA to a smart contract using the FromSLAToSmartContract library (a brief explanation of the FromSLAToSmartContract library is presented in Section 6.2.3) (see Figure 6.2).

- Add each party that is responsible for delivering a service to the Blockchain network[4] to increase the trust among the parties. The generated smart contract is deployed on each contractual party's node,

[4] A brief explanation of the Blockchain concept is presented in Section 2.1.2.

and when the arriving, reported and monitored data violate any one of the SLA terms, they will be reported.

Fig. 6.4 Abstracted SLA monitoring using Blockchain technology

- Monitor the SLA terms and record any SLA violations.

In the SLA monitoring phase, the cooperation between the smart contract and Blockchain plays a significant role in monitoring the SLA without a need for a third party in the middle. If there is a violation of one or more of the SLA terms, then one or more of the agreed-upon SLA violation actions, such as SLA termination, SLA compensation, sending a notification or reconfiguring, is applied.

- SLA Enforcement: This phase includes (a) mechanism/s that enforces the SLA terms to avoid/minimize SLA violations by applying the following and other possible SLA-aware mechanisms: mechanisms for resource allocation, resource provisioning, task placement and task scheduling.
- SLA Termination: This is the phase in which the service subscription is ended for one or more of the following reasons or other reasons: the negotiation fails to reach an agreement, one of the terms of the SLA that allows for SLA termination is violated, and the SLA reaches its end date.

6.4 Proof of Concept

In this section we aim to show how, starting with specifying the SLA, considering both SLO constraints and configuration parameters in machine readable format can be utilized in different phases of the SLA management framework:

6.4.1 Use Case Study: Flood Monitoring and Prediction (FMP)

The framework is proposed as one possible solution to ensure that SLA violations are monitored, violations are recorded and to demonstrate that our SLA specification tool can play a part within more than one phase of the proposed framework (e.g., SLA monitoring). Furthermore, the first step within the proposed framework is "workflow activity mapping", which is followed by generating the SLA specification using the SLA specification tool (the tool is presented in Chapter 5). The generated SLA is, then, utilised in the other phases of the proposed framework. Thus, we believe that the framework can be used, to some extent, for different use-cases, since its phases utilise the SLA specification tool. The tool is built based on the conceptual model and the proposed grammar, the generality and expressiveness of which we evaluated in previous chapters. Furthermore, most of participants considered different use-cases derived from their research of interests when they evaluated the tool (as discussed before in Chapter 5).

In previous chapters, we considered RHMS as a use-case scenario that we referenced for illustration purposes. However, in this section, we consider another use-case scenario, the FMP use-case. In FMP, at the application level, a simplified SLA might be "If I pay X dollars (e.g., $100) to a Cloud provider-hosted big data processing platform (see Figure 6.5), the provider must ensure that events such as road closures and bridge collapses are detected from real-time streams of social media, mobile data and historical flood modelling within a few milliseconds and that alerts are sent to the public and emergency responder teams within Y (e.g., 5) minutes of the detection". While big data analytics workloads now require guarantees in application-level SLAs, public or private Cloud providers provide only resource-level SLAs (e.g., Amazon EC2 promises 99.99% availability for its CPU, storage and network resources)[5].

[5] https://aws.amazon.com/compute/sla/

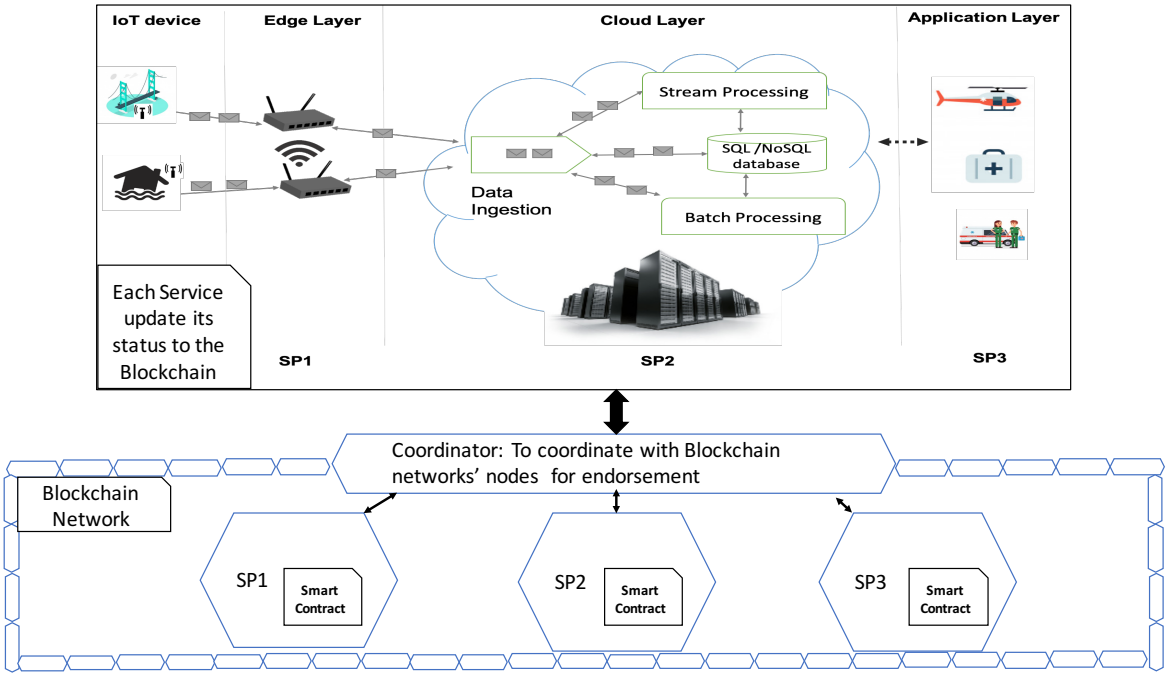

Fig. 6.5 Flood monitoring and prediction (FMP) case study

6.4.2 Implementation

For an organization that is responsible for providing an FMP service, in the early stages of running the service it is important to consult experts who can provide advice regarding where to deploy certain services and the possible configuration parameters that can aid with delivering the right action within the right time constraint. In other words, this use-case scenario can be mapped to its required workflow activities, the possible deployment strategy and the related configuration metrics. Therefore, seeking a recommender system that can be used as a second opinion to support, enhance and/or guide the expert team's decisions is one of the possible solutions. Considering the main phases included in the proposed SLA management framework, we perform the following steps:

1. Workflow Activity Mapping: Identifying the main workflow activities of FMP is the first step. FMP requires the collection of real-time data from different types of resources, such as sensors and gauges that measure rainfall levels and river water levels, respectively. The newly collected data are then analysed for any abnormal data patterns (e.g., flood possibility) by comparing them with historical/stored data.

2. Mapping each workflow activity to its required service: From step one, each activity is mapped to its required service; for example, the capture event of interest is mapped to the sensing service, and the real-time analysis activity is mapped to the stream processing service.

3. Utilize the IoT-CANE to obtain a recommended configuration parameter for each required service: Owing to the multi-layered nature of the IoT application, specifying the resource configuration to run each of the activities is a non-trivial task. Thus, we have utilized the IoT-CANE to provide us with a configuration recommendation based on some context constraints. Therefore, to identify a suitable configuration requirement, we can specify some contextual knowledge about each cooperative activity, for example, stream processing, which can be run on an Edge resource (see Figure 6.6). Then, the IoT-CANE processes the data by applying a ripple-down rules (RDR) method to recommend possible configuration parameters.

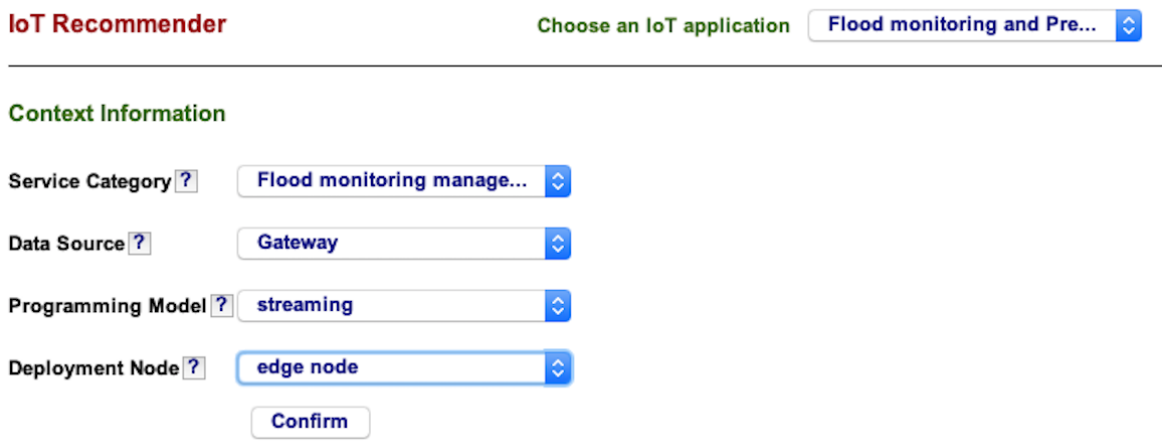

Fig. 6.6 A screenshot of the IoT-CANE screen to fill in some details of the stream processing service and recommend the configuration requirements of the service

Thus, for each activity, we ran the IoT-CANE recommender to provide a recommended configuration in a JSON format. Listing 6.1 reflects a snippet of the recommended configuration for a stream processing service that is required to "analyse small-scale real-time activity".

```
1
2 "slo" : [ {
```

```
3         "qosMetric" : "Availability",
4         "priority" : "high ",
5         "requiredLevel" : "greater than ",
6         "value" : "99 ",
7         "unit" : "% "
8       },{
9        "qosMetric" : "CPU Utilization",
10        "priority" : "high ",
11        "requiredLevel" : "greater than ",
12        "value" : "80 ",
13        "unit" : "% "
14      } ],
15      "configurationRequirement" : [ {
16        "configurationFeature" : "storage Size",
17        "value" : "99 ",
18        "requiredLevel" : "greater than ",
19        "unit" : "TB "
20      }, {
21        "configurationFeature" : "Storge Type",
22        }
```

Listing 6.1 A snippet for the recommended configuration for a stream processing service that is required to "analyse real time activity"

4. Generate SLA: Then, the created configuration is used to aid the process of specifying an SLA for FMP by running the SLA specification tool, where we can specify SLOs as well as the configuration requirements for each activity (consult Chapter 5 for more details regarding how the SLA is generated).

5. Negotiate SLA: A negotiation is conducted between the contractual parties to agree on the SLA terms. However, at this stage, we assume that the generated SLA is the one that has been agreed upon.

6. Establish SLA: Assuming the contractual parties have agreed, the SLA is now activated and established.

7. Create SLA-based Smart Contract: We create an SLA-based smart contract using the "FromSLAToSmartContract" Java library. Figure 6.7 reflects a

snippet of the generated SLA using the SLA specification tool and the corresponding snippet of the generated smart contract after using the "FromSLAToSmartContract" Java library.

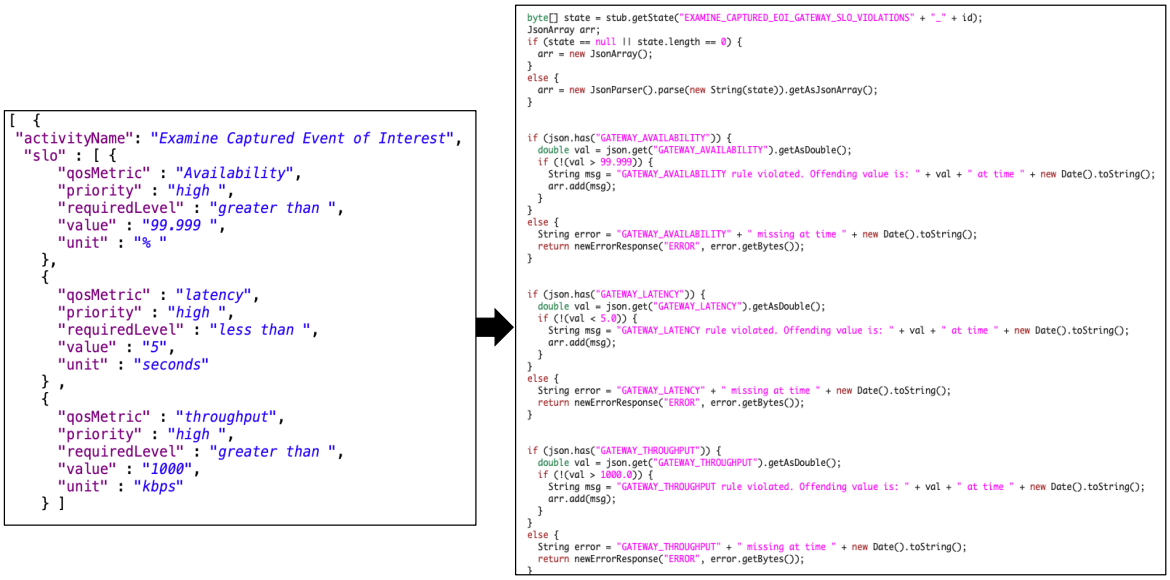

Fig. 6.7 From SLA in JSON format to SLA-based smart contract

8. Create a Blockchain network: After generating an SLA-based smart contract, we create a Blockchain network with three parties. To create a Blockchain network, we follow the "first-network" example provided by [154] to deploy a Hyperledger Fabric network for a consortium of three organizations. Each Edge resource provider (SP1), Cloud resource provider (SP2), and IoT application provider (SP3) represents a node within the Blockchain network that represents the consortium members. They participate in the administration of the Blockchain as well as in the validation and agreement. After specifying the SLA terms and converting it to an SLA-based smart contract by utilizing the library, they each have the same copy of the smart contract, and they are each updated on the state of the smart contract as it is executed, as presented in Figure 6.5.

9. SLA Monitoring: Thus, the created smart contract is deployed on each node, and each node should reflect its status based on which service it is responsible for; if there is a violation, then the parties are notified and the violation recorded. We run the following cases to update the state of a

"gateway that is an Edge resource" when performing the "examine capture EoI" activity based on what has been reported. The updating state has two cases:

- case 1: "updating without a violation"
  (a) We updated one of the gateway's SLOs, where recorded availability equals 100%, which equals the specified availability within the SLA. Latency is still within the required level (less than 5 ms)

  (b) To report the data newly acquired through monitoring, we show the update method using the command in Figure 6.8

```
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/
gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/
example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem -C testchannel -n testcontract --peerAddresses
peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/
hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/
peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051
--tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt --
peerAddresses peer0.org3.example.com:11051 --tlsRootCertFiles /opt/gopath/src/
github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/
peers/peer0.org3.example.com/tls/ca.crt -c '{"Args":
["examine_captured_eoi_gateway_slo_update","{\"GATEWAY_AVAILABILITY\": 100,
\"GATEWAY_LATENCY\": 0.4,\"GATEWAY_THROUGHPUT\": 1000, \"SOURCE_ID\": 1}"]}'
```

Fig. 6.8 Reporting the monitored SLOs of the "examine capture EoI" activity when no violation is reported

  (c) We run the command depicted in Figure 6.9 to check the violation status, and it returns no violation recorded.

```
peer chaincode query -C testchannel -n testcontract -c '{"Args":["get_examine_captured_eoi_gateway_slo_violations","1"]}'
```

Fig. 6.9 Check the violation status of the monitored data

- case 2: "updating with a violation"
  (a) In this case, we updated one of the gateway's SLOs where recorded availability equals 80%, which is less than what has been agreed upon within the SLA,

To report the data newly acquired through monitoring, we show the update method using the following command depicted in Figure 6.10:

```
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/
gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/
example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem -C testchannel -n testcontract --peerAddresses
peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/
hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/
peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051
--tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt --
peerAddresses peer0.org3.example.com:11051 --tlsRootCertFiles /opt/gopath/src/
github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/
peers/peer0.org3.example.com/tls/ca.crt -c '{"Args":
["examine_captured_eoi_gateway_slo_update","{\"GATEWAY_AVAILABILITY\": 80,
\"GATEWAY_LATENCY\": 0.4,\"GATEWAY_THROUGHPUT\": 1000, \"SOURCE_ID\": 1}"]}'
```

Fig. 6.10 Reporting the monitored SLOs of the “examine capture EoI” activity where a violation is reported

(b) We run the command depicted in Figure 6.11 to check the status of the reported violation.

```
root@ce0b43e6fcd1:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chain
code query -C testchannel -n testcontract -c '{"Args":["get_examine_captured_eoi
_gateway_slo_violations","1"]}'
["GATEWAY_AVAILABILITY rule violated. Offending value is: 80.0 at time Sat Sep 2
```

Fig. 6.11 Check the violation status of the monitored data

Figure 6.12 shows the functionality provided as part of the smart contract to update some metrics and reflect the violation status.

```
}
if (method.equalsIgnoreCase("get_latest_capture_eoi_activity_update")) {
  return get_latest_capture_eoi_activity_update(stub, params.get(0));
}
if (method.equalsIgnoreCase("get_capture_eoi_activity_violations")) {
  return get_capture_eoi_activity_violations(stub, params.get(0));
}
if (method.equalsIgnoreCase("examine_captured_eoi_gateway_slo_update")) {
  return examine_captured_eoi_gateway_slo_update(stub, params.get(0));
}
if (method.equalsIgnoreCase("get_latest_examine_captured_eoi_gateway_slo_update")) {
  return get_latest_examine_captured_eoi_gateway_slo_update(stub, params.get(0));
}
if (method.equalsIgnoreCase("get_examine_captured_eoi_gateway_slo_violations")) {
  return get_examine_captured_eoi_gateway_slo_violations(stub, params.get(0));
}
```

Fig. 6.12 A snippet of functionality provided within the generated smart contract to update some metrics and reflect the violation status

10. SLA enforcement: This is the phase that ensures that the SLA terms are respected by providing SLA-aware mechanisms for resource provisioning, scheduling, management, service placement, etc. SLA enforcement is left as a future work.

11. SLA Compensation: Whenever there is an SLA violation, then the corresponding SLA compensation should be applied; however, SLA compensation is left as a future work.

12. SLA termination: Termination typically occurs in one or more of the following cases: the negotiation ends with no agreement, there is a violation of the SLA terms that involves SLA termination or if the contract reaches its end date/time.

6.5 Discussion

In the previous section, we implemented a prototype in which we followed the stages of the proposed framework. We first mapped the FMP use-case scenario to its workflow activities and associated each activity with the required service. Then, for each service, we ran the IoT-CANE to obtain the recommended configuration parameters, which were then used to aid the process of specifying the SLA. However, for the SLA negotiation stage, we assumed only that the specified SLA is one that we can base the agreement on. Then, the generated SLA was converted to a smart contract using the "FromSLAToSmartContract"

library. After that, the SLA-based smart contract was created and deployed on a Blockchain network to allow the monitoring stage to start. The monitoring stage was performed with mimicked data, and the smart contract recorded all violation cases.

We test the generated smart contract after deploying it on different nodes on the Hyperledger Fabric network. We test the chaincode and the network to show that the generated-deployed smart contract could detect all the SLA violations. Instead of using a real FMS system, we utilised an emulator, which is developed by [85], to send transactions to the blockchain networks. The emulator is created with the programming languages NodeJS and JavaScript. The emulator takes advantage of JavaScript's asynchronous programming model so that multiple transactions can be submitted without waiting for responses. Thus, the simulator will send transactions to the blockchain network in fast succession. The emulator includes fields configurable by the users to model different structures of the IoT system with sensors, gateways, real-time analysis services, etc. For each element of the IoT application, the emulator will send state update transactions. For instance, if the user configures the emulator to emulate an IoT application consists of 100 sensors on the IoT device layer, the Fabric network will receive 100 sensor update-state transactions to emulate the system deployed.

The emulator not only includes fields that allow the user to simulate various infrastructures; it also includes a 'VIOLATION RATIO' field. 'VIOLATION RATIO' field is a number between 0 to 1 which controls the number of elements that can cause a violation. When an emulator is configured by a user which consists mainly of 100 sensors in the IoT device layer with a breach ratio of 0.05, for instance, 5 sensor states with an SLA breach are submitted by iteration.

The emulation logic is provided in a loop such that the user can set up the emulator to run X iterations. State transactions for each element in the IoT topology are submitted to the emulator. For instance, if the user has configured the emulator to consist of 100 sensors, 50 gateway devices, 5 ingestion services, 3 real-time analytics, and 2 storage services, thus SLO/Requirement state update transactions will be executed for all 160 elements per iteration. The number of submitted transactions depends on the number of updates on the SLO/Require-

ments state.

The use of the previously mentioned emulator allowed the network and generated smart contract to be evaluated. We mimic different violation cases that represent data related to the FMP SLA. A breach ratio of 0.05 was used for each test with at least one element per layer causing a breach. Every test was performed for three iterations. These tests aimed to verify that the generated-deployed smart contract could detect all the SLA violations generated by the emulator and that each submitted state was reported in the ledger. The smart contract succeeds in reporting and recording the SLA violation using Blockchain technology (see Figure 6.11). The findings show a 100% success rate in capturing all the emulated violations related to the FMP use case. Table 6.1 shows the number of cases with different infrastructures and services.

Table 6.1 Number of Detected Violated Cases

No Sensors	No Gateway	No Ingestion Service.	No Real-time Analysis Service.	No Storage Service.	No Transactions	No Detected out of total number Violated cases
1	1	1	1	1	17	15/15
2	1	1	1	1	18	17/17
200	50	1	1	1	317	29/29
1000	600	5	5	5	3460	173/173

SLA enforcement and compensation are left as future work.

6.5.1 Comparison with Other SLA Management Frameworks

We have compared our proposed SLA management framework with a number of studies [411, 278, 378, 474, 257, 464, 72, 466] that were selected because they are closely related to the proposed framework.

The research presented in [411] introduces a framework to effectively track and evaluate the SLA parameters and to detect the possibility of SLA violations occurring [411]. The proposed framework considers the SLA negotiation and monitoring phases. It also introduces an adaptive resource allocation method to avoid SLA violations by utilizing the results of SLA violation prediction results. The negotiation is presented only within the proposed framework. The resource allocation mechanism is applied as a mechanism to enforce the SLA by avoiding or at least minimizing SLA violation cases.

Patel et al. [378] propose a method for managing SLAs in a Cloud computing environment using the WSLA framework built in a service-oriented architecture (SOA) for SLA monitoring and enforcement. They use the WSLA's third-party support function to perform monitoring and enforcement to overcome trust issues. SLA specification and negotiation are part of the proposed framework, but they are outside the scope of this study.

Uriarte et al [474] propose a framework for SLA management by relying on the rich and complex formalism of an SLA that is converted into a smart contract. This contract is then managed via the Blockchain. Landi et al [257] propose an architecture and some protocols for managing SLAs of mobile Cloud networking services.

Developing an easy-to-deploy and easy-to-use Cloud platform and creating an effective monitoring tool to monitor SLA violations on the application layer is not only valuable to consumers but also vital to Cloud providers. Thus, Liu et al [278] propose an SLA management approach that monitors the SLA to detect any SLA violation at the application layer.

Torkashvan and Haghighi [464] suggest a Cloud SLA framework called CSLAM, which has a language for presenting the agreements between the signatory parties. In addition, it has a mechanism for the deployment and monitoring of the SLA parameters. They claim that CSLAM is an applicable framework, and they delegate its implementation to future research.

Benedictis et al [72] introduce an SLA management solution that is supported by a REST API. The implementation that they provide for the proposed REST API supports the SLA processes of negotiation, renegotiation, monitoring and applying the associated actions whenever a violation occurs.

Touloupou et al [466] seek to link business requirements with observable attributes as SLOs between network operators and customers. During SLA template generation, they allow network operators to choose between different SLOs and then automatically formulate an agreement based on those choices. Finally,

they provide a monitoring system for detecting any breaches and alerting users.

To compare our proposed SLA management framework with the above-mentioned related studies, we have considered certain criteria that are related to some SLA lifecycle phases. The selected criteria are related to the scope of our solution and are not conclusive:

- Specification: this criterion is considered to be fully covered if the study considers SLA specification to be a phase within its SLA management framework and applies it using, for example, a prototype or a real system. However, if the study considers SLA specification as simply a phase within its SLA management framework, with no implementation provided, then the SLA specification can be classified as partially covered.
- Negotiation: this criterion is considered to be fully supported if the study considers SLA negotiation and implements it using a prototype or within a real system use-case scenario. If the study just proposes it as a phase within its SLA management framework, then the criterion can be described as partially supported.
- Monitoring: this criterion is considered to be fully covered if SLA monitoring is part of the SLA management framework and is implemented. If a study just proposes SLA monitoring as a phase within its SLA management framework, then the criterion can be identified as partially covered.
- Enforcement: This criterion is assumed to be fully supported if the research considers SLA enforcement and applies it using, for example, a prototype or use-case scenario. If the study simply proposes SLA enforcement as a process within its SLA management structure, the criterion can be identified as partially fulfilled.
- Recommender System: If the study considers and implements a recommender system, this criterion is considered to be fully supported. If the research proposes only a recommender system phase within its framework with no implementation, then the criterion is partially covered; otherwise, it is not covered at all.

- Blockchain for trustworthiness: The criterion is fully covered if the study considers Blockchain technology and smart contracts and applies them for management purposes. If the study only proposes Blockchain technology and smart contracts within its framework, then the criterion is partially covered; otherwise, it is not covered.

Table 6.2 Comparison of the SLA management frameworks. Black circles represent features fully supported in the framework, empty circles represent a partially supported feature and a hyphen (-) means not supported.

Reference	Specification	Negotiation	Monitoring	enforcement	Blockchain	Application area
[411]	○	○	●	●	-	Cloud
[278]	-	-	●	-	-	Cloud
[378]	○	○	○	○	-	Cloud
[474]	○	○	●	●	●	Cloud
[257]	○	○	○	○	-	Cloud
[464]	●	●	●	○	-	Cloud
[72]	-	●	●	-	-	Cloud
[466]	●	-	●	-	-	5G for Network Services
Proposed Work	●	○	●	○	●	IoT

Table 6.2 shows a comparison between a number of available SLA management frameworks and the proposed framework. Most of the frameworks have been developed for the Cloud computing paradigm, except [466], which focuses on G5 for network services. However, our proposed framework is the only one that focuses on IoT applications. Considering that the integration of Blockchain technology and smart contracts for trust purposes is still in its early stages, Uriarte et al [474] utilize the integration of Blockchain technology and smart contracts. However, they monitor the SLA based on their previous SLA specification, which focused on the Cloud infrastructure tier. Furthermore, the aim of their work is to enhance trustworthiness and eliminate/minimize the need for a third party for Cloud services. However, our approach considers an end-to-end SLA for the IoT and it therefore considers the most common services and infrastructures resources across layers.

6.6 Conclusion and Future Research

In the background section of this chapter, we present two implementation applications that utilize the SLA specification tool. In the first case, since the tool provides fine-grained details of the configuration parameters, it is used to create the knowledge base of a Cloud/Edge-based context-aware recommendation system for IoT configuration purposes. In the second case, it translates the generated SLA from a JSON format to a smart contract. Then, these two applications became part of our proposed SLA management framework, which consists mainly of the following phases: context-aware recommender, SLA specification, SLA negotiation, SLA-based smart contract for SLA monitoring using Blockchain after establishment and SLA enforcement.

There are limitations to our proof-of-concept implementation, including SLA negotiation, and SLA compensation and enforcement, which have not been applied; therefore, they remain as areas for future research. Furthermore, the transition from one step to another is done manually, although it can be done autonomously by developing an agent-based SLA management framework. Therefore, in future research, one possible avenue is to combine what has been developed (i.e., the SLA specification tool, context-aware recommender and SLA-

based smart contract) and add other artefacts for the missing functionalities that will facilitate an autonomous SLA management framework.

Chapter 7

SLA-aware Approach for IoT Workflow Activities Placement Across Layers

Overview

Previous chapters (Chapter 3, 4 and 5) discuss the SLA specification for the IoT and Chapter 6 proposes the SLA management framework. The proposed SLA management framework consists mainly of the following phases: context-aware recommender, SLA specification, SLA negotiation, SLA-based smart contract for SLA monitoring using Blockchain after establishment, and SLA enforcement. In this chapter, we focus on SLA enforcement policies, since the chapter considers the management of resources to reduce economic penalties arising from potential SLA violations [343]. Therefore, we propose an SLA-aware approach for IoT workflow activity placement across layers. The decision over whether activities are assigned to the Edge or the Cloud to enforce the SLA is based on a number of specified constraints within the SLA, such as constraints associated with each activity and the configuration parameters. Thus, we define the problem (Section 7.2.1) then propose a greedy heuristic algorithm to allocate activities between the available resources across layers while minimising the execution time (Section 7.2.1). The allocation algorithm considers factors such as the deadline associated with each activity, location and budget constraint. We evaluate the proposed work using iFogSim for three use-case studies (7.3). The results show a reduction in cost, time, and energy consumption compared

with the Cloud-only placement approach and the Edge-ward placement algorithm.

7.1 Introduction

With the increasing number of applications and their time-sensitive nature, Cloud-based solutions are not enough and they commonly suffer from latency due to the centralized nature of Cloud-based data centres, which are mostly located far from the data sources [372, 284]. Therefore, utilizing Edge resources while benefiting from the Cloud whenever required is essential, especially for time-sensitive applications and to overcome the problems associated with centralized control [372, 284]. As a result, strategies for planning and the efficient allocation of resources that consider multiple SLA parameters are needed [150].

On the one hand, in order to escape expensive penalties, SLA violations should be avoided, while on the other hand, providers should use resources effectively in order to reduce service provisioning costs [150]. Thus, it is necessary to consider SLA-aware approaches that include, but are not limited to, SLA-aware resource allocation, SLA-aware provisioning, SLA-aware activity/task placement and SLA-aware scheduling. For example, Cloud service provisioning can be performed based on the specified terms within the SLA, which include non-functional service conditions defined as QoS, and responsibilities and penalties in the event of contract breaches [150].

This chapter proposes an SLA-aware approach for workflow activity placement across Cloud and Edge-computing layers. The approach considers the nature of the IoT where various nodes generate vast quantities of records, and data processing solutions consist of a number of activities/tasks that can be executed at the Edge of the network or on the Cloud. Management at the Edge of the network may limit the time required to complete responses and return the final results/analytics to the end users or applications. Additionally, IoT nodes have a restricted amount of functionality over the contextual information gathered owing to their restricted computational and resource capacities.

A number of works propose service placement mechanisms. For example, Tran et al [468] offer a new, multi-layered, IoT-based fog computing architecture. In particular, they develop a service placement mechanism that optimizes service decentralization in the fog landscape by using context-aware information such

as location, response time and service resource consumption. The approach is being used in an optimal way to increase the efficiency of IoT services in terms of response time, and energy and cost reduction. However, this study considers tasks to be independent, which is not the case with IoT applications. Furthermore, it is not an optimal approach since it applies the constraints match approach.

Naas et al [348] seek to take advantage of fog nodes' heterogeneity and location to reduce the overall latency of storage and data retrieval in the fog. They formulate the data placement problem as a generalized assignment problem and propose two solutions: 1) an exact solution using integer programming; and 2) a geographically based solution to decrease the solving time. However, their focus is on storing data at the Edge to ease data retrieval, i.e. related to the database tier only.

Kolomvatsos and Anagnostopoulo [139] propose a placement approach that takes into account the power consumption of a system and minimizes delay violations using a discrete particles swarm optimization (DPSO) algorithm. iFogSim is used to evaluate the proposed approach. Kolomvatsos and Anagnostopoulo [243] suggest a smart decision-making system to assign tasks locally, but they consider only the effect of their algorithm on energy consumption and delay. Furthermore, this approach allows for task processing to be executed only in sequential order (there is no parallel execution capability).

In this chapter, we propose a greedy heuristic algorithm to allocate tasks between the available resources while minimizing the execution time. However, there are a number of challenges to consider, such as maximizing the utilization of Edge resources while considering the limitations of their computation capabilities. Additionally, there is a possibility that some of the tasks will be time sensitive, and it is therefore crucial that they be allocated and executed immediately. Executing forthcoming tasks requires proper task allocation and scheduling that satisfies the requirements of all the tasks while maintaining the SLA. Therefore, we propose a layer-based algorithm that identifies and minimizes the global bottleneck, i.e., minimizing the processing time and the cost as well as maximizing the utilization

of resource computation at the Edge layer. The main contributions of this chapter are summarized as follows:

- We consider a task placement approach based on cooperation between the Edge and the Cloud that supports service decentralization, leveraging context-aware information such as the location, and the available computing and storage capacities of Edge resources. The placement approach considers the execution time constraint associated with each task and the budget constraint at the application level. This maximizes the utilization of Edge resources and minimizes latency, energy consumption and cost.
- We conduct a performance analysis with various case studies using iFogSim[1] to reveal the effectiveness of the proposed approach in terms of maximizing the utilization of fog devices while reducing latency, energy consumption and network load.

7.2 SLA- and context-aware approach for IoT activity placement across the Cloud and the Edge

In this approach, we consider IoT applications, which consist mostly of a set of activities, some of which require high bandwidth and low computation. They can be performed on one of the resources at the Edge of the network, while if an activity requires more computation, it can be offloaded to the Cloud. In a case where there is more than one activity, the selection of which one to deploy at the Edge or the Cloud level can be based on different criteria (e.g., cost, location, processing time or processing capacity). Our aim is to provide a solution that distributes the workflow activities in a way that respects the consumer's requirements and utilizes the computation capabilities across layers while aiming to avoid any SLA violations.

[1] iFogSim is an open-source toolkit for modelling and simulating resource management approaches to the IoT and Edge and fog computing, https://github.com/Cloudslab/iFogSim.

7.2.1 Problem Definition and Modelling

In IoT applications, task/activity/inter-module [2] placement is the problem of allocating tasks/activities to a set of processors/resources/infrastructure resources [3] that are distributed across layers (Edge and Cloud). The input into the task placement controller is an activity graph and a processor graph, and the output is a placement plan that maps each activity to a suitable processor/resource. Whether the resources are located at the Edge or the Cloud is based on each task's/activity's computation and communication requirements.

The following describes the task placement information and the concepts related to the proposed scheme.

Task Graph:

A task graph is represented by a directed acyclic graph (DAG), $TG = (T, E)$, where the set of nodes $T = \{t_1, t_2, t_3, ..., t_n\}$ represents n tasks. Between tasks, there is a set of edges belonging to E, which represents the data dependency between nodes. For example, tasks t_1 and t_2 are connected by $e_{1,2}$. In other words, between any t_i and t_j, there is $e_{i,j}$ belonging to E. Thus, we can define edges and tasks as follows:

$$\forall (t_i \wedge t_j \in T),\ \exists\, e_{i,j} = (t_i, t_j)\ \in E$$

Consider that T represents a task that is defined as T = {T_{id}, ReqPCapcty, deadline, region, level}. T_{id} represents task Id, $ReqPCapcty$ represents requested processing capacity, $deadline$ represents the deadline constraint of the time execution of a single task, and $region$ reveals the region/location where this task is preferably deployed. Finally, $level$ is used to denote how many hops this task takes from the starting point, which in our case is sensing events.

Each $e_{i,j}$ can be defined as $e_{i,j}$ = ($SrcID$, $DisID$, DTR, TRP, TupleLentgth), where $SrcID$ represents the source task id (t_i node), $DisID$ represents the destination task id (t_j node), DTR expresses the data transfer rate between t_i and t_j, TRP represents the processing requirement of coming tuples, and TupleLength represents the total length of the tuple.

[2] Within the text, we use task, workflow activity and inter-module interchangeably.
[3] Within the text, we use processor, resource and infrastructure resource interchangeably.

Each task has one or more predecessors, unless if it is a start task, and each task has one or more successors unless if it is a finish task (see Figure 7.1, which depicts the start and finish tasks). Any task starts only after all the predecessor tasks have been completed, so the earliest start time of a task is equal to the maximum finish time of all of its predecessors.

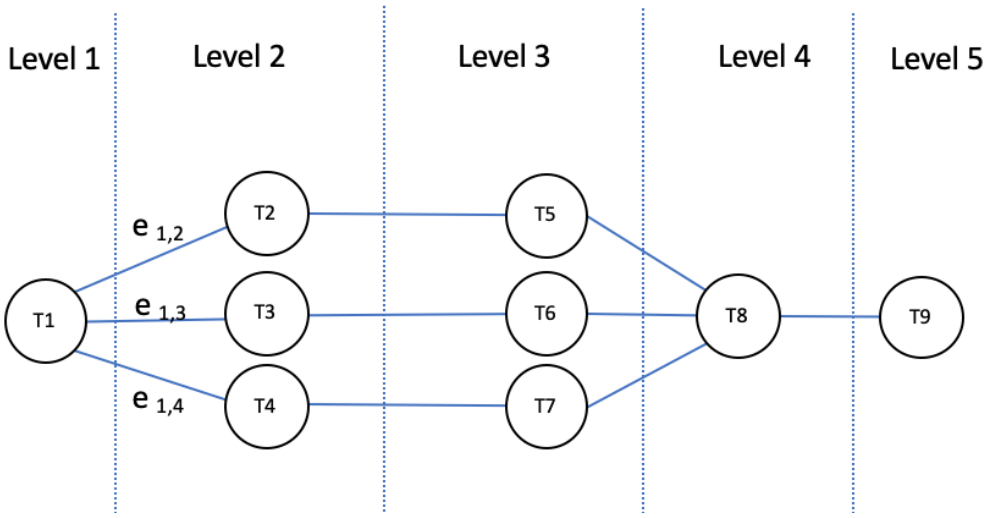

Fig. 7.1 Task dependency example for an IoT application

Processor Graph

Consider the topology presented in Figure 7.2. A processor graph is represented by a DAG, $PG = (P,D)$, where the set of nodes $P = p_1, p_2, p_3, ..., p_n$ represents n processors. A processor belonging to P can be a Cloud or an Edge resource. Between processors is a link distance, d, that connects them; for example, processors p_1 and p_7 are connected by $d_{1,7}$, so there is a set of links $d_{i,j}$ between any p_i and p_j belonging to P and $d_{i,j}$ belonging to D. We can define the distance links and processors as follows:

$$\forall p_i \wedge p_j \in P, \exists\, d_{i,j} = (p_i, p_j) \in D\,; \qquad p_i\ AND\ \ p_j\ \ are\ \ not\ \ leaf\ nodes$$

Each processor p_i can be defined as p_i =($pCapcty_i$, $upLnkLatency_i$,$downLnkLatency_i$, $pmLoad_i$). $pCapcty_i$ is considered to hold the processing capacity of p_i, up link latency is $upLnkLatency_i$, down link latency is $downLnkLatency_i$ and $pmLoad_i$ represents the current PM load.

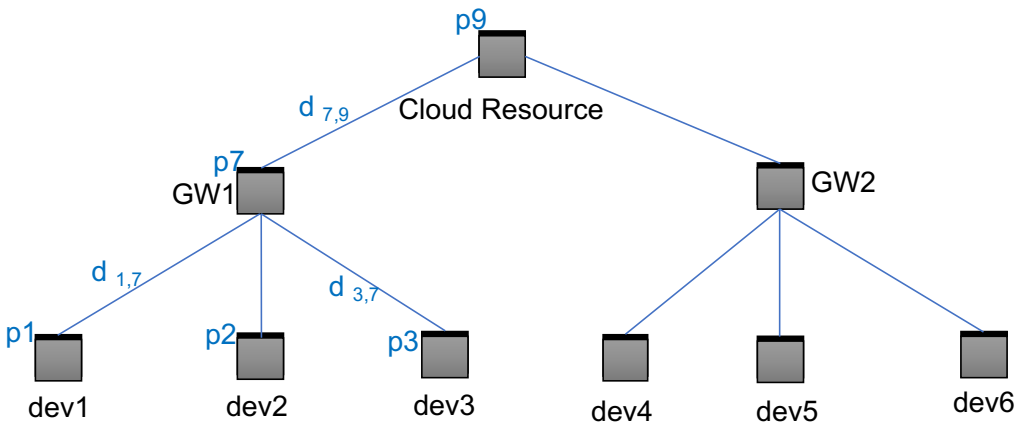

Fig. 7.2 Processor graph

Objectives

The objective is to propose a placement mechanism that aims to maximize the utilization of Edge resources and minimize the cost and execution time of an IoT application to adhere to defined SLO constraints at the application level. In this study, the main information considered for task/workflow activity placement is as follows:

- Network Topology: available resources and their computation capabilities.
- Location: the location of the initiated requests or consumed services.
- Service Type: data storing and data filtering are services that require different computation/storage capabilities; therefore, the approximate size of the data (e.g., million instructions per second (MIPS)) required by an activity/task is different based on the type of required service.
- Level of Activity: number of hops that separate a task/activity from its starting point, in our case from the IoT devices that generate the data. It is essential to denote the dependency between tasks, as this helps to avoid assigning a task to a resource on a level lower than its predecessor, unless the predecessor requires higher computation resources. It also allows parallel processing for tasks that are on the same level.
- Quality of Service: advance knowledge of the constraints on the offered services plays a role in selecting the type and layer of resource. In our work,

we consider minimizing the end-to-end response time by considering the deadline constraint for each involved task/activity.

To express our objective of maximizing the utilization of Edge resources, each task t_i can be deployed on R_{Cloud} (Cloud resource) or on R_{Edge} (Edge resource); however, deploying t_i on a Cloud or Edge resource is a binary variable. If t_i is deployed on the Cloud, then 1 is assigned to $T_{i_{R_{Cloud}}}$ and 0 to $T_{i_{R_{Edge}}}$, and vice versa.

Each task is processed on either Edge or Cloud resources; thus, we try to maximize the number of tasks that are assigned to Edge resources whenever appropriate, which we represent mathematically as:

$$Maximize \sum_{i=0}^{i=n} T_{i_{R_{Edge}}} \tag{7.1}$$

The processing time of a task t_i on a resource p_j is calculated as shown in equation 7.2.

$$T_{Exec}(t_i, p_j) = \frac{\alpha_{p_j}^{t_i} f_{t_i}(z_i)}{p_j} + upLnkLatency_{p_j} \tag{7.2}$$

Here, $f_{t_i}(z_i)$ represents the computation requirement for task t_i, z_i represents data input into task t_i and $\alpha_{p_j}^{t_i}$ represents the number of current modules running concurrently with t_i on node p_j.

The calculation of the CPU requirement for upcoming data/tuples for task t_i is given in equation 7.3, for each Edge has t_i as its distention (i.e., the DisID for Edge e is t_i). DTR represents the data transfer rate of Edge e, and TPR represents the required processing capacity for each tuple transferred by Edge e.

$$\sum_{Edgee=0}^{Edgee=y} DTR \times TPR \ \forall \ Edge\ e\ has \ \ DisID \ = \ t_i \tag{7.3}$$

The total cost of running t_i over node p_j is the sum of the memory cost $memCost_{Ti}$, the communication cost $commCost_{Ti}$, the storage cost $storgCost_{Ti}$ and the node cost $nodeCost_{Ti}$, as given in equation 7.4. Each of these costs is explained in equation 7.5-7.8. (Note: memoryCostUnit represents the cost defined per memory unit; commCostUnit means defined cost per communication unit; StorageCostUnit represents the cost defined per storage unit; sizeDataIn is the

size of the coming data; DataToStore is the size of the data to be stored and NodeCost represents the cost of using a node.)

$$Cost(T_i) = memCost_{Ti} + commCost_{Ti} + storgCost_{Ti} + nodeCost_{Ti} \tag{7.4}$$

$$memCost_{Ti} = \frac{f_{t_i}(z_i)}{p_i} * memoryCostUnit \tag{7.5}$$

$$commCost_{Ti} = sizeDataIn * commCostUnit \tag{7.6}$$

$$storgCost_{Ti} = DataToStore * StorageCostUnit \tag{7.7}$$

$$nodeCost_{Ti} = T_{iExec} * NodeCost \tag{7.8}$$

Proposed Algorithm

Proposing a multi-objective SLA-aware algorithm for placing workflow activities across layers is essential, subject to the constraints of each involved service/infrastructure resource as specified within the SLA. Yet, in this early-stage work, we only considered the end-to-end execution time and a number of constraints related to the involved services/infrastructure resources. For example, we considered the cost and execution time at the application level as well as the time constraint associated with each required service and the capacity capabilities of the required infrastructure resource. In the following section, we present the proposed algorithm for placing inter-modules among the available resources while considering the following objectives and constraints:

Offline Integer Programming Formulation Here, we aim to minimize the cost of deploying inter-modules on available resources and the end-to-end response time. For the offline version of the inter-module placement problem, integer programming formulations are derived. These formulations are used to devise limits on the suggested approach. The main objective is to minimize the execution time while considering other constraints, such as cost/budget constraints. Table 7.1 summarizes the notations used in our formulation. Our main decision variables, denoted as x_{ij}, are defined as follows:

$$obj = \textbf{Minimize} \sum_{i=0}^{i=n} Time_{t_i} \tag{7.9}$$

subject to

$$x_{ij} \in 0,1, \forall i = 0,1,...,n; \forall j = 0,1,...,m \tag{7.10a}$$

$$\sum_{i=0}^{i=n} Cost_{T_i} <= CostConstraint \tag{7.10b}$$

$$\sum_{i=0}^{i=n} T_{i_{R_{Edge}}} > \sum_{i=0}^{i=n} T_{i_{R_{Cloud}}} \tag{7.10c}$$

$$\sum_{i=0,level=l}^{i=n} t_{ij} <= 1 \tag{7.10d}$$

$$\sum_{k=0,i,j}^{k=predecessorListSize} t_{kj_{tier}} <= t_{ij_{tier}} \tag{7.10e}$$

Constraint 1 enforces the binary nature of x_{ij}. Constraint 2 ensures that the cost of the deployment plan will not exceed the cost/budget limit (CostConstraint). Constraint 3 ensures that the number of inter-modules assigned to Edge resources is greater than the number of inter-modules assigned to Cloud resources, as far as possible. Constraint 4 ensures that no two inter-modules from the same level are assigned to the same resource. Constraint 5 ensures that no inter-module t_i is assigned to a resource that is located in a tier lower than any of its predecessors (considering only predecessors of t_i that require processing capacity less than the processing capacity required by t_i).

In Algorithm 1, the task and resource graphs are input. Lines 6, 7 and 8 define the associated level of each inter-module, which is calculated based on the number of hops between the inter-module and the source of the captured interesting event. Lines 10 to 12 calculate the corresponding execution time of deploying t_i on resource p_j and then sorting all resources based on their execution time. Starting with the shortest execution time, we need to apply the constraint-checking steps, which are in Lines 14 to 28: these steps check whether the p_j resource has predecessors to ensure that no predecessor of inter-module t_i is assigned to a resource that is allocated to a layer higher than the current checked p_j if it requires more computing capacity than its predecessors.

If there is a resource p_j that has one of t_i's predecessors deployed on it, and no predecessor of t_i is deployed on a resource that is located in a layer above the current p_j's layer, then the associated constraints are checked with inter-module t_i by calling the checkConstraintsConsistencyFunction (Line 22). If there is no

Table 7.1 Notations for the Offline Integer formulations and symbols used in the algorithm

Notations	Meanings
t_i	task/activity/intermodules i
p_j	a resource with processor pj
e_{ij}	dependency edge between two tasks t_i and t_j
l_{ij}	Link between two resources Pi and Pj
bw_{ij}	bandwidth of link l_{ij}
z_{ij}	data size transferred over edge dependency e_{ij}
dis_{ij}	distance between Pi and Pj
d_{ij}	Link delay of link l_{ij} between Pi and Pj
$p_{i_{capacity}}$	Computation capacity of resources Pi
$ReqPCapcty$	Represents requested processing capacity
$level$	Reflects how many hops this task is far from the starting point
$e_{i,j}$	Edge between t_i and t_j task nodes
$SrcID$	Represent the source id
$DisID$	Represents the destination id
$dataTransfereRate$	Expresses data transfer rate between t_i and t_j
$TupleProcessingReq$	Represent processing requirement of the tuples
$TupleLentgth$	Represents the length of tuples
$pCapcty_i$	Holds processing capacity of p_i in MIPS
$upLnkLatency_i$	Up Link Latency of p_i
$pLoad_i$	Current CPU load of p_i
$TupleLentgth$	Represents the length of tuples
$T_{i_{R_{Edge}}}$	Task t_i is running on Edge resource
$T_{i_{R_{Cloud}}}$	Task t_i is running on Cloud resource
$TimeConstraint_{t_{ij}}$	Time constraint of running t_i
$Time_{t_{ij}}$	Execution Time of running t_i on resource p_j
$t_{ij_{tier}}$	Reflects the tier of resource p_j that runs t_i
$predecessorListSize$	predecessors of t_i with $ReqPCapcty$ less than t_i's
$SortedResorces$	Sort resources based on their execution time of task t_i in ascending order

Algorithm 1: SLA aware algorithm for application module placement across layers

```
Input: TG=(T,E) ; PG=(P,D)
region =-1
found= false
Output: application modules are mapped to available resources
Objectives: Minimize Cost and Minimize Application latency
foreach t_i in TG do
    define a deadline d constraint
    assign a level value to t_i
    // level for each task reflects how many
    // hops between the task and the starting point
    foreach resource p_j in PG do
        calculate Time_{t_ij}
        add(SortedResorces, p_j)// sort resources based on their execution
            time of task t_i in ascending order
    end
    foreach resource p_j in SortedResorces do
        if t_i has a predecessor then
            foreach t_k in predecessor list of t_i do
                if t_k is already assigned to a PG resource then
                    p_l = the PG resource that t_k is assigned to
                    if t_kReqPCapcty is less than t_iReqPCapcty then
                        if p_llevel is greater than p_jlevel then
                            break
                else
                    call checkConstraintsConsistencyFunction
                    if checkConstraintsConsistencyFunction then
                        assign t_i to p_j update list of assigned modules of p_j
                          Found= true
                    else
                        continue
                    end
                end
            end
    end
    if not found then
        Calculate the Cost of executing t_i on Cloud as in Eq. 7.4
        if TotalCost+=cost of executing t_i on a Cloud resource is less than
          budget then
            update TotalCost
            assign t_i to a Cloud resource
    else
        Log the Cost exceeds the allowed Budget and break
    end
end
```

Algorithm 2: Checking Budget Constraints Consistency after mapping tasks to resources

```
1  checkConstraintsConsistencyFunction
2  if region==-1 then
3      if t_iregion == p_jregion then
4          if that p_j does not have tasks from the same level of t_i then
5              Calculate the Execution Time of t_i on p_j by applying Equation 7.2
6              if the requested CPU Less than available CPU then
                   // Check if the resource can sustain the place module
7                  if the time is less than or equal to the associated deadline
                     with t_i then
8                      return true
9          else
10             change the region value from -1 to another region not the same
                 as region of t_i return false
11         end
12 else
13     Go To line 4
14 end
```

resource p_j that matches the requirements and considered constraints, then t_i is assigned to a Cloud resource. The checkConstraintsConsistencyFunction checks that a resource p_j can sustain the inter-module within its deadline constraint, without exceeding the budget, that it is not running other tasks at the same level as the coming inter-module and that it is within the same region. If searching all resources within the same region does not satisfy the constraints, then the other regions are checked; otherwise, false is returned. If false is returned, then the task is assigned to the Cloud resource, provided that the cost constraint is not violated.

7.2.2 Time Complexity Analysis

We solved this problem with a context-aware approach. If the inter-module does not have predecessors, then in the best-case scenario, the first search attempt returns a resource that matches the requirement for each inter-module; thus, the time complexity is $\theta(n)$, where n represents the number of inter-modules. However, in the worst-case scenario, when finding a resource that matches the inter-module constraints and performing it for each inter-module t_i, all m

resources in the resource list must be checked. Therefore, the time complexity is $\theta(nm)$, where m represents the number of IoT devices (e.g., mobiles) in the available resources in PG. Cases where an inter-module t_i has predecessors require more time, in order to avoid assigning an inter-module to a resource in a layer lower than the predecessor resource's layer if it requires more computing capacity than its predecessors. Thus, we perform a checking step that iterates all of an inter-module's predecessors. As a result, in the best-case scenario, when an inter-module has only one predecessor and finds a resource that matches the inter-module constraints at the first attempt, the time complexity is $\theta(n)\theta(1)$, and considering only the upper bound means that the time complexity equals $\theta(n)$. In the worst-case scenario, when an inter-module t_i has k predecessors, then finding a resource that matches the inter-module constraints leads to all available m resources being checked. In this case, time complexity can be calculated as $\theta(nmK)$, which is the worst-case scenario for time complexity: $\theta(nm)+\theta(nmK)$; however, since k, which represents the number of predecessors of an inter-module, is less than the total number of inter-modules n, the time complexity is $\theta(nm)$.

7.3 Evaluation

To evaluate our proposed algorithm, we ran it using the iFogSim simulator.

iFogSim

The iFogSim simulation toolkit is based on CloudSim [99] platform [306]. CloudSim is one of the most widely utilized cloud-based simulators [70, 304, 306]. iFogSim provides a wide scope for modelling a customized Fog computing environment with such a large number of Fog nodes and IoT devices (e.g. sensors, actuators) by extending CloudSim classes. However, iFogSim annotates the classes to make the service and infrastructure resource allocations policies for Fog computation easily defined for users without any previous knowledge of CloudSim. iFogSim uses Sense-Process-Actuate and distributed data flow models when simulating any Fog-computing application scenario. There are a number of simulators for the IoT; however, we chose iFogSim because it is based on CloudSim, which is popular among researchers for testing various strategies/algorithms, such as

[468, 348, 139]. In iFogSim, the measurement of end-to-end latency, network congestion, energy use, operating costs and QoS satisfaction is supported. Furthermore, in a substantial number of research projects, iFogSim has been used to mimic resources [459, 306], latency [308, 306], Quality of Experience (QoE) [309, 306], mobility [80, 306], energy [303, 306], QoS-Aware management of the Fog computing environment [431, 306] and security[104, 306]. It simulates IoT applications, where it can enable application modules to be allocated and scheduled among Fog and Cloud resources. It also provides two case studies to explain IoT modelling and resource management policies which we utilized to evaluate our proposed approach.

7.3.1 Use-Case Studies

For evaluation purposes, we consider the following case studies:

Remote Health Monitoring Service (RHMS) Case Study 1

Consider a remote health monitoring service (RHMS) to which patients (e.g., elderly patients with Parkinson's disease) can subscribe in order to be monitored on a daily basis. Data are collected and filtered, and if there is a pattern of interest or an event that matches a threshold value, then the data can be analysed on a small scale as they are related to a specific patient. However, in cases that require a comparison between incoming data and historical data, or in cases where the same events come from different subscribers, such as when many patients have signs of fever and many social media users are tweeting about these widespread signs, then the scenario can be considered as a large-scale data analysis task that needs high computational power. The most interesting analysis results are then stored.

In this use case, the workflow activities – data collection, data filtering, small-scale real-time data analysis, large-scale real-time data analysis and storing data – are represented as tasks t_1, t_2, t_3, t_4, and t_5, respectively. There are sensors attached to patients' wrists, video cameras for patients and mobile accelerometers to capture activity patterns. These devices are connected to smart phones as a gateway, which is then connected to the WiFi gateway. The

WiFi gateway is connected to an Internet service provider, which in turn is connected to a Cloud data centre [4].

Intelligent Surveillance Case Study 2

The intelligent surveillance application [5] comprises five main processing modules. 1) Motion Detector reads the camera's raw video stream to detect an object's motion. 2) Object Detector extracts objects from the video stream, and if an object is new compared with previously discovered objects that are currently active in the area, then object tracking is activated. It also determines the object's coordinates 3) Object Tracker determines the optimum pan-tilt-zoom (PTZ) configuration based on the last calculated coordinates of the objects currently being tracked. These PTZ data are regularly transmitted to the PTZ control of the cameras. 4) PTZ Control is embedded in each smart camera and it changes the physical camera configurations to suit the optimum PTZ parameters sent by the tracker module. 5) User Interface is an interface that displays to the user a segment of the video streams that contain each tracked object. This case study is one of the case studies mentioned in [186]. We use it because we plan to compare our results with the Edge-ward module placement algorithm presented in [186]. For more details of the case study and the algorithm, readers are advised to refer to [186]. Listing 7.1 shows a snippet of the SLA of the case study 2 [6].

```
1
2  [  {
3     "appType" : "Intelligent Surveillance",
4     "startDate" : "Wed Nov 21 00:00:00 GMT 2018",
5     "endDate" : "Thu Nov 21 10:35:46 GMT 2019",
6     "Execution Time" less than "1000 milliseconds"
7     "Cost/Price" less than "1000.0 $ per contract period"
8      "activityName" : "Capture Event of Interest: Motion Detector "
9      "Sample Rate" greater than  "3 kHz "
10     "deviceType" : "Sensor "
11     "numberOfDevices": "4 "
```

[4] The generated SLA for case study 1 is listed in Appendix B
[5] Consult [186] for further details, look here [186].
[6] The generated SLA for case study 2 is listed in Appendix C.

```
    "mobilityOfDevice": "fixed "
    "communicationTechnology":"WiFi "
    "cpuCapacity":"1.6 GHz"
    "memorySize": "1 GB"
    "activityName" : "Examine Captured EoI: Object Detector",
    "Network Latency" less than to "2 milliseconds "
    "Size of data-in" greater than "4 KB "
    "Link Bandwidth" equal to "6 Kbps "
    "deviceType": "Gateway "
    "numberOfDevices": "4 "
    "mobilityOfDevice":"fixed "
    "communicationTechnology": "WiFi "
    "cpuCapacity" "3 GHz"
    "memorySize": "4 GB"
   "activityName" : "Analyse large-scale realtime data:     Object
     Tracker  ",
    "Latency" less than  "5 milliseconds "
    "Memory Size" : " 1 KB "
    "vCPU Capacity": "2 GHz Xeon "
    "Hypervisor": "Xen "
    "OS Type" "Linux Ubuntu "
}]
```

Listing 7.1 A list of some of the considered constraints/ configuration for case study 2

EEG Beam Tractor Game Case Study 3

The EEG Beam Tractor Game [7] comprises three main processing modules. 1) The client receives the raw EEG signals, checks for any inconsistencies in the received signal values and excludes any potentially inconsistent readings. 2) The Concentration Calculator calculates the brain status and concentration level of players using the sensed EEG signal values. To update the state of the player, the client module is informed of the level of calculated concentration. 3) The Coordinator operates globally to coordinate the game among multiple players

[7] Consult [186] for further details of the EEG Beam Tractor Game.

who may be active in locations that are geographically separated [8]. This case study is one of the case studies mentioned in [186] because we plan to compare our results with the Edge-ward module placement algorithm presented in [186]. For more details of the case study and the algorithm, readers are advised to refer to [186].

7.3.2 Physical network

For the case study, we considered a physical topology with different types of Fog devices. Table 7.2 and Table 7.3 present the configuration of the topology used. This configuration is the same for both case studies except for the number of IoT devices. Case study 1 consists of four areas, and each area has four IoT devices. Case study 2 consists of two areas, and each area has four IoT devices. Case study 3 consists of four areas, and each area has six IoT devices.

Table 7.2 Associated latency of network links

Source	Destination	Latency [ms]
IoT device	Smart Phone	1
Smart Phone	WiFi Gateway	2
WiFi Gateway	ISP Gateway	2
ISP Gateway	Cloud Data Centre	100

Table 7.3 Configuration description of infrastructure resources

Device Type	CPU [GHz]	RAM [GB]
Smart Phone	1.6	1
WiFi Gateway	3	4
ISP Gateway	3	4
Cloud VM	3	4

7.3.3 Performance Evaluation Results

Analysis of Case Study 1

We applied the proposed algorithm to case study 1 and compared the performance result with placing the inter-modules on the Cloud. Execution time, energy

[8] The generated SLA for case study 3 is listed in Appendix D.

consumption, network usage and cost are the metrics that are captured by simulating the application and applying the proposed approach to place the inter-modules using iFogSim. The following subsections compare the results of applying the proposed approach with those of applying the Cloud-only approach.

Execution Time The overall execution time for case study 1 was less for the proposed approach for task placement than for the Cloud-only approach (Figure 7.3). The latency control loop is reflected in Figure 7.4. "In iFogSim, the developer can specify the control loops to measure the end-to-end latency" [186].

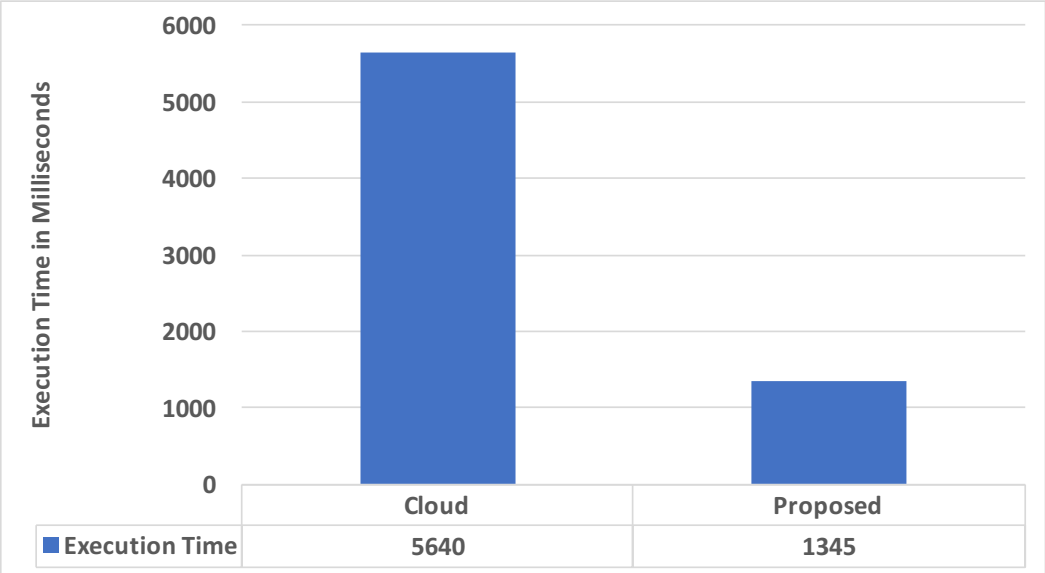

Fig. 7.3 Time execution of case study 1

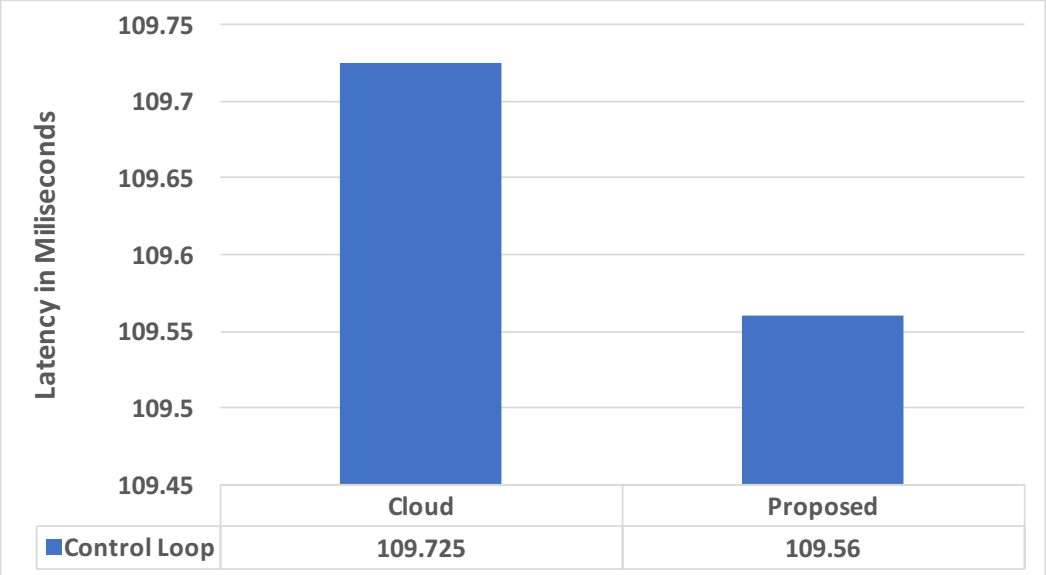

Fig. 7.4 Control loop delay in case study 1

Network Usage As shown in Figure 7.5, there is not much difference in network usage; however, the proposed approach shows slightly more network usage at the Edge, probably because it allocates most inter-modules to the Edge resources.

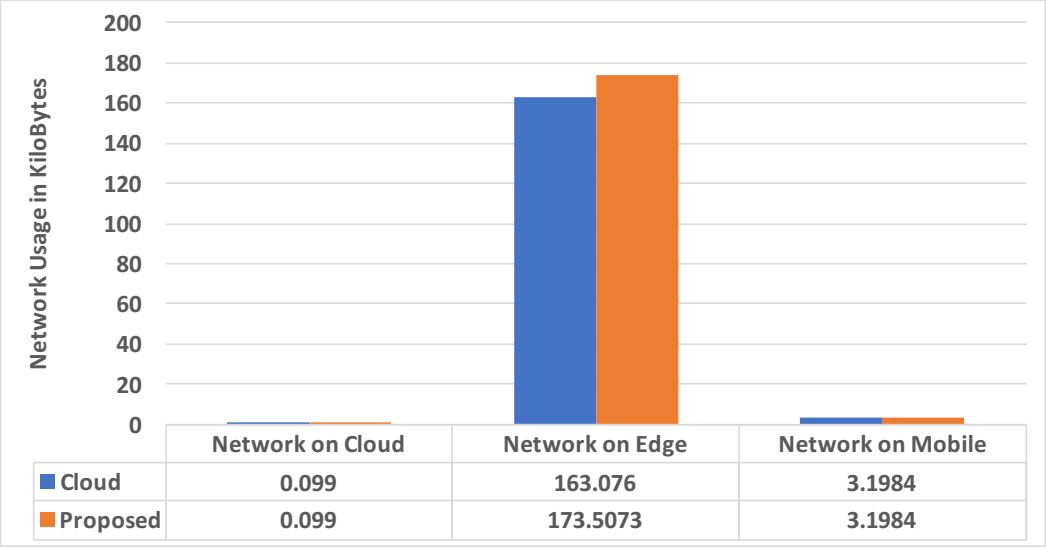

Fig. 7.5 Network usage of case study 1

Energy Consumption In general, the Cloud-only approach, as depicted in Figure 7.6, showed a higher level of energy consumption on both the Cloud and

IoT device layers than the proposed approach. In the Edge layer, there was a slight difference between the two approaches, possibly for the same reason, which is that the proposed approach allocates more inter-modules to the Edge than to the Cloud.

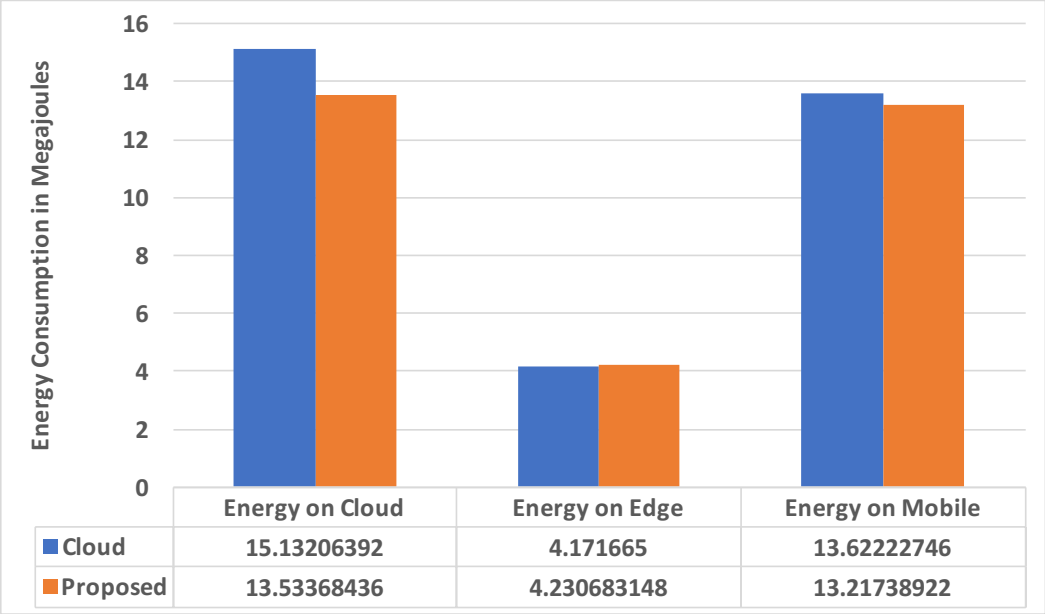

Fig. 7.6 Energy consumption of case study 1

Cloud Cost Figure 7.7 depicts the cost of implementing case study 1 when applying the Cloud-only approach and our proposed approach. The Cloud cost is higher with the Cloud-only approach, while our approach is five times less costly than the Cloud-only approach because the proposed approach allocates more inter-modules to the Edge than to the Cloud.

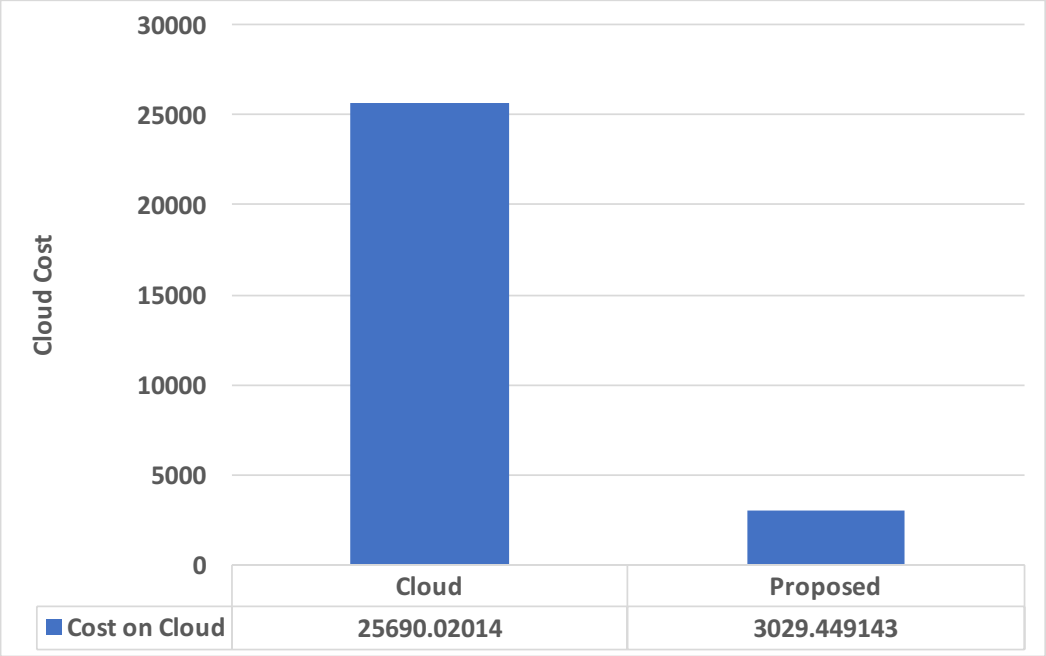

Fig. 7.7 Cloud cost of case study 1

Analysis of Case Study 2 and Case Study 3

iFogSim comes with two built-in case studies that apply the Edge-ward inter-module placement approach proposed in [186]. Therefore, we applied the proposed algorithm to case study 2 and case study 3 and compared the performance results with those obtained when placing the inter-modules with the Cloud-only approach, as well as with the Edge-ward inter-module placement approach. The metrics of execution time, energy consumption, network usage and cost were captured. The following sections describe the comparison results.

Execution Time The execution time of all three approaches (Cloud-only, Edge-ward placement and our proposed placement) is presented in Figure 7.8 for both case studies 2 and 3. In case study 2, the Edge-ward placement approach showed a high level of execution time, while in case study 3, the Cloud-only placement had the highest level of execution time. However, in both cases, our proposed approach had the shortest execution time.

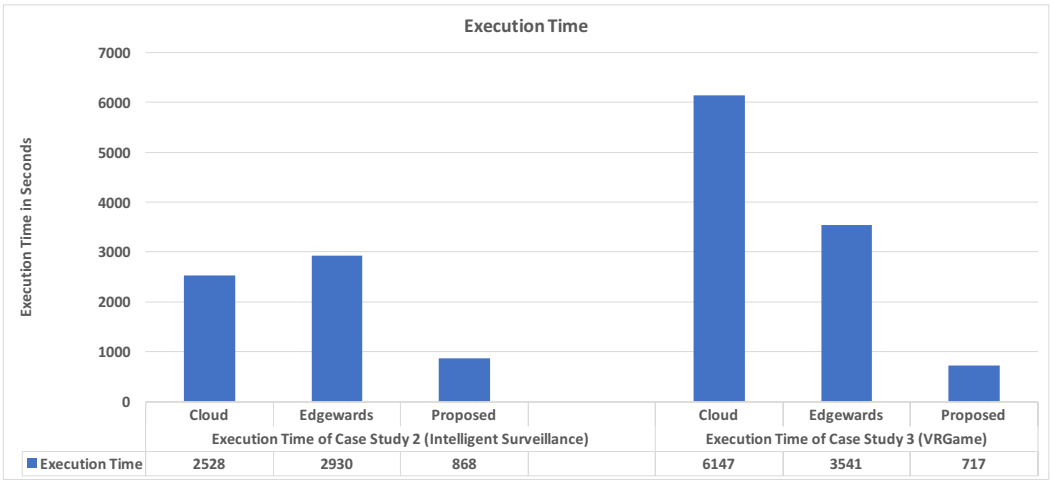

Fig. 7.8 Time execution of case study 2 and case study 3

Network Usage Network usage is measured for the three tiers: IoT devices (mobile phones), Edge resources (WiFi gateways) and the Cloud. Network usage in case study 2 was higher on the Edge than on the Cloud and mobile tiers when applying the Cloud-only placement approach. It reached 0.099 KB, 326.0964 KB, and 6.3968 KB for the network on the Cloud, the Edge and IoT devices, respectively. The Edge-ward placement reflected the least network usage among all the proposed approaches: the Cloud (0 KB), the Edge (16.4832 KB) and IoT devices (0.3968 KB). The proposed approach showed no network usage on the Cloud but it showed a high level of network usage on the Edge tier (1038.61192 KB).

Case study 3 shows that there was higher network usage for the Edge than for the Cloud and IoT devices when applying the Cloud-only placement approach. It reached 114.87 KB, 151.1156 KB, and 2.1202 KB on the Cloud, Edge and IoT devices, respectively.

When we applied the Edge-ward placement, it resulted in the lowest network usage among all the proposed approaches: the Cloud (0 KB), the Edge (2.3976 KB) and IoT devices (2.3952 KB).

When applying the proposed approach, there was no network usage on the Cloud, but a high level of network usage was recorded for the Edge tier (121.1936 KB), where mobile devices reached 2.3319 KB. The reason for this may be that

tasks were placed on different Edge resources since this approach considers parallel processing for independent tasks.

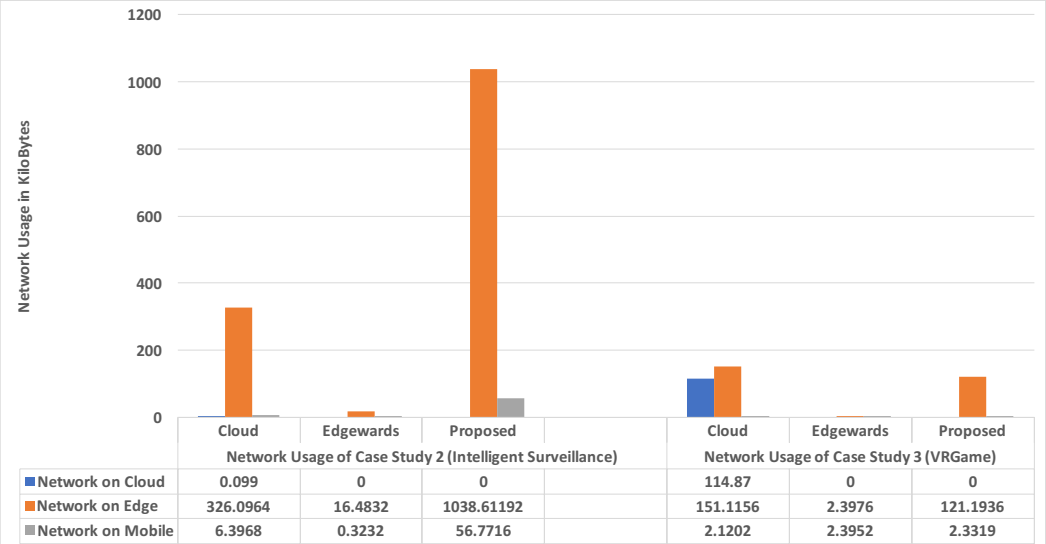

Fig. 7.9 Network usage of case study 2 and case study 3

Energy Consumption Figure 7.10 presents the energy consumption of the three approaches for both case studies 2 and 3. For case study 2, the Cloud-only approach resulted in the highest energy consumption on the Cloud tier(14.2812 megajoules); however, on the Edge and mobile tiers, all approaches reflected quite similar levels of energy consumption, with a slightly lower level for the proposed approach on the Edge and mobile tiers. On the Cloud, Edge and mobile tiers, energy consumption was 13.3497 megajoules, 2.5029 megajoules and 6.9441 megajoules, respectively, for the Edge-ward placement approach. On the Cloud, Edge and mobile tiers, energy consumption was 13.32 megajoules, 2.5639 megajoules and 6.6069 megajoules, respectively, for the proposed placement approach.

For case study 3, the Cloud-only approach resulted in the highest energy consumption on the Cloud tier (15.0633 megajoules); however, on the Edge and mobile tiers, all approaches reflected quite similar levels of energy consumption, with slightly lower levels for the proposed approach on the Edge and IoT devices. On the Cloud, Edge, and mobile tiers, energy consumption was 13.32 megajoules, 3.5501 megajoules and 10.5032 megajoules, respectively, for the Edge-ward placement approach. On the Cloud, Edge and mobile tiers, energy consumption

was 13.32 megajoules, 3.3373 megajoules and 9.9699 megajoules, respectively, for the proposed placement approach.

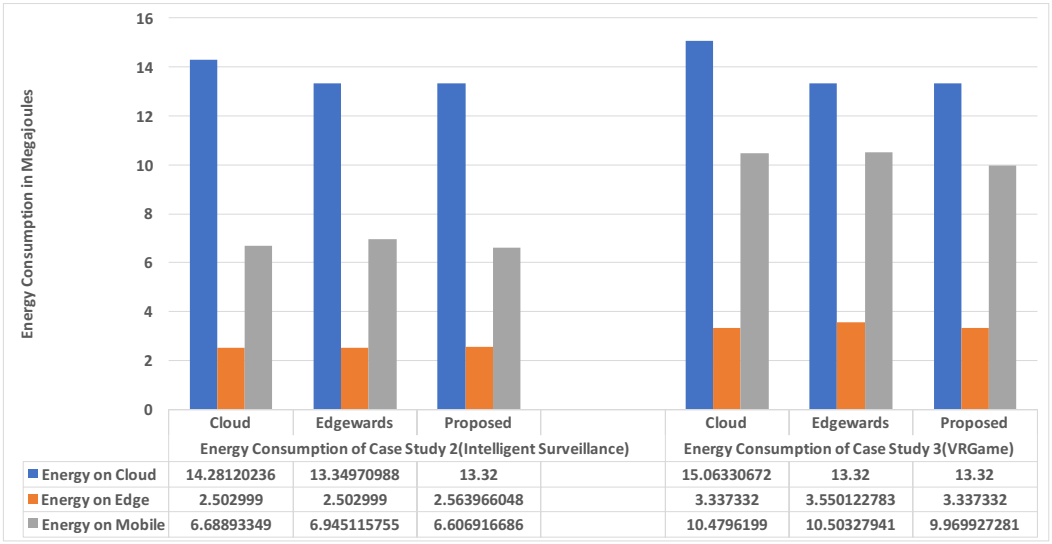

Fig. 7.10 Energy consumption of case study 2 and case study 3

Cloud Cost Figure 7.11 shows the Cloud cost for deploying use cases 2 and 3. Since the proposed approach placed all tasks on the Edge, there was no Cloud cost in either case study. The Edge-ward placement approach had a lower cost than the Cloud approach in case study 2. In case study 3, the Edge-ward approach Cloud cost was null because all inter-modules were placed on the Edge.

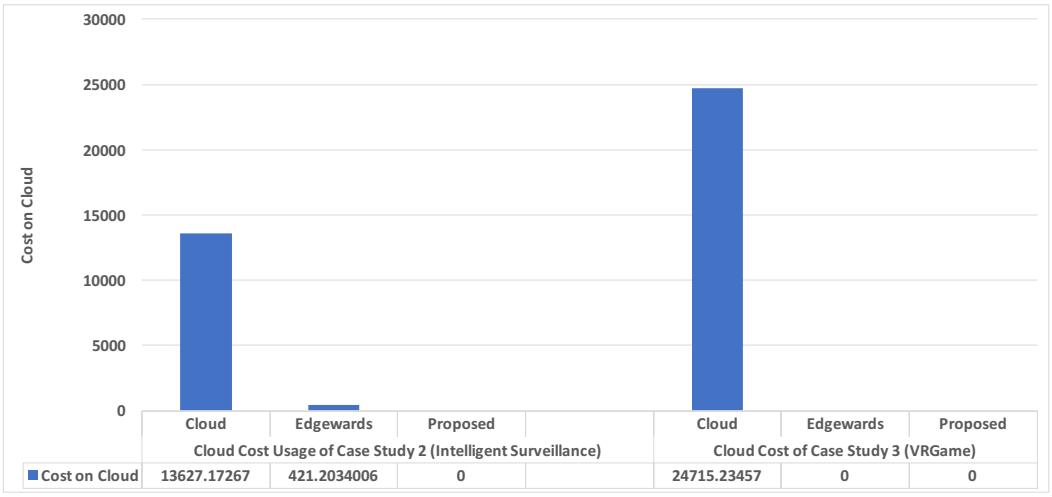

Fig. 7.11 Cloud cost of case study 2 and case study 3

7.4 Discussion

We have applied our heuristic algorithm to decentralize task placement in a cooperative way between Edge and Cloud resources. We have considered an RHMS as a case study and compared it with the Cloud-only approach (an approach in which tasks are placed only on the Cloud). The proposed approach demonstrates lower execution time, control loop delay, costs, network usage and energy consumption than the other approaches. Furthermore, we considered comparing our approach with built-in use cases in iFogSim. Thus, we compared the results of our algorithm with the Edge-ward placement algorithm applied to case study 2 and case study 3. In general, our proposed approach shows better results than the other approaches, as the previous section showed.

In the proposed approach, in case study 1, when there were some inter-modules that required higher processing capabilities, such as for storing data and analysing large-scale data, these were placed on the Cloud. This was why there was a Cloud cost as well as network usage on the Cloud layer. In the other two case studies (case study 2 and case study 3), all inter-modules were placed on the Edge layers, which explains the null cost and network usage on the Cloud layer. However, in case studies 2 and 3, our proposed approach showed a higher level of network usage on the Edge layer than the Edge-ward placement approach, which seems to be related to the fact that our approach ensures that independent tasks are placed on different resources; thus, there is a greater possibility of more data being transmitted between resources since this approach considers parallel processing for independent tasks.

7.5 Conclusion and Future Research

Due to the limited computational and resource capabilities of IoT nodes, tasks can be allocated to Edge or Cloud resources, taking into account a number of factors such as task constraints, node load and computing capability. We suggested a heuristic algorithm for allocating tasks among the available resources. The allocation algorithm takes into account factors such as the related time limits for each task, location and budget constraints. We utilized iFogSim to evaluate the

proposed approach for three use cases. The performance analysis demonstrated that the suggested algorithm minimizes costs, execution time, control loop delay, networking and Cloud power usage compared to the Cloud-only and Edge-ward positioning methods.

In general, although the proposed approach showed good results, it is a greedy approach, which means that a better placement plan may be identified by considering all available solutions. Furthermore, in this approach, our main objective is to minimize the execution time, whereas realistically, applications have multiple objectives in addition to minimizing the execution time, such as minimizing the cost and maximizing the number of processed requests. In future research, we will carry out an evaluation of the proposed approach on real systems. Furthermore, we need to extend the proposed algorithm to apply multi-objective placement algorithms and to develop algorithms that take advantage of machine learning approaches, such as applying multi-objective genetic algorithms in which we consider more than one objective, for example minimizing response time and maximizing throughput.

Chapter 8

Conclusion and Future Research

Overview

In this chapter, we summarise the research presented in this thesis. We outline our contributions and then discuss open research problems on the grounds that they could influence future research.

8.1 Thesis Summary

Many service providers, including Cloud providers, provide a written description of the terms and conditions of their services. This introduces some drawbacks, such as confusion and no facility for automating the evaluation of adherence to SLA terms, searching for facilities or negotiating contract terms. This highlights the importance of providing SLAs in a machine-readable format. Moreover, the importance of machine-readable agreements is also demonstrated by the need for structured assurances that the services delivered comply with the terms agreed, as Cloud consumers, for example, may outsource their core business functions to the Cloud.

Many languages have been proposed to describe SLAs in a machine-readable format in order to simplify their evaluation and negotiation. However, available SLA frameworks range from over-specific to over-generic. We argue that these languages cannot cope with the distinctive features of the IoT, such as multi-party IoT agreements and multi-layer deployment models. In IoT applications, QoS specifications need to be aggregated from the Cloud, network, and sensing layers'

perspectives. The key aim of considering QoS across layers is to deliver the promised IoT functionalities that match the expectations of consumers, as agreed within the SLA.

Since this research focuses on SLA specifications for the IoT, we conducted a systematic review to classify most of the IoT SLA-related research. We researched the SLA lifecycle by analysing the literature in order to shed light on the available SLA-related works. However, due to the limited work on SLAs for the IoT, previous works, especially Cloud-related works, were also considered due to the IoT-Cloud interdependence. We mapped 400 papers from various scientific databases. We defined two key categories that most SLA-related research falls into: SLA-lifecycle work and work related to SLA applications. The results showed that around two-thirds of papers concentrate on the SLA lifecycle: SLA specification, SLA negotiation, SLA monitoring, SLA enforcement and SLA management. The remaining papers focus on SLA applications such as SLA-aware resource allocation and scheduling. We have tracked the growth in SLA research and discussed a range of research gaps to be considered in future studies. There was a shortage of studies relevant to SLAs for the IoT in general and a shortage of studies that contribute to SLA specifications, in particular. Therefore, in Chapter 2, we presented a reference architecture for the IoT as a first step towards SLA specification is defining the key concepts considered in our proposed conceptual model presented in Chapter 3.

An important feature of an SLA language is the possibility of expanding it to include domain-specific vocabularies. Therefore, a domain SLA language needs to consider domain-specific support features. Thus, in Chapter 3, we suggested an IoT conceptual SLA model as well as rich domain-specific vocabularies to express an SLA of the IoT on an end-to-end basis. The proposed conceptual model defines key concepts that can play a role in the formulation of SLAs on an end-to-end basis. Then, we defined some of the most common QoS metrics and configuration parameters related to each concept. We referred to the RHMS as a use-case scenario for illustration purposes when we explained some of the concepts. We believe that the proposed vocabularies can play a significant role in SLA specifications, particularly with regard to the standardisation of the vocabularies/terminologies used in an SLA. We applied a goal-oriented approach to

evaluate the proposed SLA conceptual model. The Wilcoxon test was used, to examine whether there was a significant satisfaction with the conceptual model's generality and coverability. The outcome indicates a high level of satisfaction regarding the generality and coverability of the considered concepts.

Defining "SLA offers" and "SLA requests" using standard terminology facilitates the process of comparing the available choices and choosing the most appropriate consumer-based SLA bid. Moreover, it is important to provide the SLA specification in a machine-readable format as a first step towards automating SLA monitoring and management. However, we believe that current SLA specification formats are insufficient to accommodate the unique features of the IoT domain, such as its multi-party multi-layer nature. In addition, most available SLA specifications are defined in an XML format, whereas we are looking at more lightweight SLAs for the IoT. Furthermore, available works are defined for other computing paradigms such as Cloud and web services, while in the IoT, there is a need to consider cooperated services that are, commonly, spanned across layers. Thus, with the necessity of delivering a high-quality service considering the strict constraints at the application level for the IoT, there is also a need to consider SLAs on an end-to-end basis. Chapter 4 presented a grammar for the syntactic structure of the end-to-end SLA specification for IoT applications. The structure of the syntax is derived from the conceptual model proposed in Chapter 3.
We evaluated the proposed grammar using the GQM approach to reflect its generality and expressiveness. Furthermore, we provided qualitative analysis for the missing requirements. Based on the percentage of overall dissatisfaction with both Goal1 and Goal2, we can see that there is a high level of satisfaction with the presented list of requirements for both goals. The requirements suggested by the participants are taken into consideration by refining the listed vocabularies as well as adding the suggested ones wherever possible.

One of the reasons behind proposing a grammar is that it is a way to unify the SLA specifications, which can then be the first step to define an SLA in a machine-readable format. Having an SLA in a machine-readable format can play a significant role in automating SLA monitoring and management and in providing SLA-aware solutions, such as SLA-aware scheduling, SLA-aware resource provisioning, SLA enforcement and SLA monitoring. In the literature, some

works consider the provision of a machine-readable SLA. For example, there are a number of different works that use XML documents to standardise SLAs in order to improve the SLA data exchange between the consumer and the provider. These works, as far as we are aware, allow users to type in or edit SLA templates. Therefore, in Chapter 5, we presented a toolkit to mitigate human typing errors when specifying an SLA using GUI features. The tool considers the most common or typical IoT application tiers and services, as captured within the proposed grammar (presented in Chapter 4), in such a way that interested users can specify their preferences. Using the tool, the SLA is generated in a JSON format. We generated the SLA in a JSON format due to the lightweight nature of JSON and its readability. We evaluated the tool following a goal-oriented approach, and the outcome of the evaluation demonstrates a high level of satisfaction with the simplicity and generalizability of the considered concepts and the tool's capability to express the requirements of involved services and infrastructure resources.

Due to the importance of SLA management as one of the solutions that can mitigate the risk of violating SLA terms, we proposed an SLA management framework. There are a considerable number of studies in the literature covering SLA management. However, there is a shortfall in the number of available and feasible SLA management frameworks that develop a management mechanism for SLA lifecycle phases (e.g., definition, negotiation, monitoring and enforcement) for IoT applications. Furthermore, there is a shortfall in the utilisation of Blockchain technology and in works that have combined the SLA specification with the recommender system, especially in terms of considering the complexity of specifying requirements with the multi-layered nature of IoT applications. Thus, in Chapter 6, we proposed an SLA management framework which consists mainly of the SLA specification, SLA negotiation, SLA monitoring using Blockchain to deploy SLA-based smart contracts, SLA enforcement and SLA compensation. In addition, we added an optional configuration recommendation phase since, as explained in Chapter 3, some configuration parameters such as sample rates can affect the data analysis accuracy because they affect the data freshness.
Thus, having a recommender system combined with the SLA specification phase provides considerable help for most contractual parties of IoT-based services. It can be utilised as a guide for most common configuration parameters for different infrastructure resources across layers. We provided a proof of concept for the

proposed SLA management framework. The main purpose of the proof of concept is to reflect how providing SLA specifications in a machine-readable format can be useful. It is not only important for the SLA specification phase, but can be utilised in more than one phase of SLA management. We discussed the proposed SLA management framework and compared it with other related studies.

In Chapter 7, we discussed the importance of using Edge resources to resolve issues relating to centralised control while benefiting from the Cloud in order to adhere the SLA. As a result, strategies are required to prepare and efficiently distribute resources, in a manner that takes into account SLA parameters. SLA violations should be avoided in order to prevent costly fines, and providers should use resources wisely in order to minimise service provisioning costs. SLA-aware techniques include, but are not limited to, SLA-aware resources allocation, SLA-aware resource provisioning, SLA-aware activity/task placement and SLA-aware scheduling. Related to SLA management, we believe that one of the possible ways to enforce/respect the SLA is to develop SLA-aware solutions such as SLA-aware scheduling and resource provisioning. Therefore, we proposed the SLA-aware workflow activity placement algorithm for IoT applications. For the proposed algorithm, we considered end-to-end execution time as the main objective while considering other constraints, such as cost and each activity deadline constraint, to aid the process of decentralising IoT activities among Edge and Cloud resources. The results show improved cost, time, and energy consumption compared with the Cloud-only placement approach and the Edge-ward placement algorithm.

8.1.1 Limitations

1. We have tried to generalise our study, but we believe that it is still necessary to identify more domain-specific vocabularies related to other workflow activities. However, the toolkit is extendable to a certain extent in that it enables interested users to add, edit and remove vocabulary terms related to services/infrastructure resources as well as to add, edit and/or remove services, infrastructure resources and workflow activities using the Excel file attached to the toolkit.

2. There is a need to consider the correlation between configuration parameters. We have specified some of the most common configuration parameters and have mentioned that there is a dependency between them, but we still need to analyse their dependency using benchmarking approaches.

3. We have considered price as a concept with parameters to specify the related constraints; however, there is a need to model the (*price*) concept in more detail and to consider different vocabularies that can be used to reflect the cost dimension for each involved service and/or infrastructure resource. For example, in the networking service, one of the cost dimensions could be link bandwidth; in stream processing, it could be data payload size; and for a Cloud VM, it could be the number of vCPUs. Furthermore, there is a need to model SLA-based compensation in the case of SLA violations from each contractual party's perspective.

4. There is still a need to consider other SLO constraints within the proposed SLA-aware algorithm, since we have considered only deadline constraints, while there are other constraints such as the throughput of requests received per time unit.

8.2 Future Research

Regarding future research, our proposed SLA management framework for IoT ecosystems is still in an early stage; therefore, we can consider the following:

8.2.1 SLA negotiation protocol to enhance consumer experience when selecting a service provider

SLA negotiation is the second phase in our proposed SLA management framework. In the current system, we assume that all contractual parties have agreed upon the specified terms. Therefore, in future research, we will investigate applying an SLA-based negotiation protocol to enhance consumer experience when selecting a service provider and allocating services and resources.

One possible protocol for SLA-based negotiation is the Contract Net protocol[1]. The Contract Net protocol [433] provides a structured high-level interaction between nodes that cooperate to deliver a task. It focuses on how negotiation can be used as a mechanism for interaction at different levels of complexity. The uses of Contract Net vary [482] and it has been utilised for the negotiation of SLAs for grid resource management and for utility computing [482].

The study in [401] presents an SLA coordination mechanism based on the market and an established Contract Net protocol. Ranjan et al [401] apply the Contract Net protocol because it allows the owners of the resource to have greater control over how resources are allocated compared to traditional mechanisms and because it creates super-schedulers who are able to bid for SLA contracts through Contract Net, with a focus on ensuring that the task is completed within the specified time frame.

According to [187], with regard to Cloud services, in order to meet the requirements of consumers, it is essential that there is sufficient collaboration between brokers and service providers. The customers' requirements should be mapped against the resources available on the Cloud, and these resources should be accessed automatically through web services. However, this type of automated service faces challenges from distributed and constantly changing Cloud-computing environments. These challenges include dynamically contracting service providers, whose service fees are determined by supply and demand, and having to cope with incomplete data regarding Cloud resources, such as providers and locations. The research in [187] aims to solve these challenges by employing an agent-based Cloud service composition approach. In this approach, the Cloud resources and participants are implemented and supported by agents who maintain a three-layered self-organising multi-agent system. This system supports a Cloud service composition framework and an experimental test bed. Networks of acquaintances and the Contract Net protocol are employed by the agents to develop and adapt Cloud service compositions.

[1] "In Contract Net, Agents acting as managers announce tasks on Contract Net, and other agents who are contractors assess these announcements; if the task interests them, they may bid for it. The outcome of a bid can be only total rejection or acceptance, so Contract Net is suited to multilateral processes such as auctions. ."[482]

8.2.2 Build cross-layer multi-provider SLA-based monitoring systems for the IoT

SLA monitoring is essential to ensure that the service is delivered according to a specified quality level. Therefore, many studies on SLA/QoS monitoring have been conducted [412, 228, 184, 116], (these references have been covered in Chapter 2). Studies such as [228, 38, 412] perform SLA monitoring in web services and at the grid and Cloud paradigms, while [184] implements a distributed monitoring system for network resources. Additionally, [116] provides monitoring as a service and supports its monitoring system by using map rules to map low-level metrics to high-level SLA requirements. However, it is applied to the data infrastructure in the Cloud. From the IoT perspective, there is still a need for end-to-end SLA monitoring; therefore, as future work, we will focus on that need.

From an IoT perspective, it is still necessary to fulfil the need to have end-to-end SLA monitoring; therefore, as future research, we will investigate the following: How can we build cross-layer multi-provider SLA-based monitoring systems for the IoT that support the end-to-end SLA adherence process of IoT applications? Answering this question is important because it will aid service providers in operating their services at an adequate level, which will then increase consumers' trust and help to avoid SLA violations. A monitoring service is used to gather data that represent the required metrics. These metrics can be used to evaluate to what extent service consumers and providers comply with the specified QoS constraints. If it is shown that the conditions of the contract have not been met, then corrective action can be taken [487]. This QoS monitoring service enables organisations to ascertain the cause of any performance issue they are experiencing, whether it be the application design, the infrastructure of the network, or the Cloud service provider [408].

Furthermore, investigating the integration of Blockchain-based solutions for SLA compensation is one of the future research avenues. SLAs contain service information, and considering a "penalty" as one of the main aspects is crucial because the consumer must be compensated accordingly if the service provider does not deliver what has been agreed upon. However, the current compensation

method is time consuming and complicated. Therefore, utilising Blockchain for SLA-based monitoring and compensation could be an enormous asset. An effort to address this problem is underway. For example, an approach based on Blockchain and smart contracts is suggested by [55] to automate the compensation process while allowing dynamic payments over the lifecycle of the SLA.

References

[1] Aazam, M., St-Hilaire, M., Lung, C.-H., and Lambadaris, Ioannis, P. (2016). PRE-Fog: IoT Trace Based Probabilistic Resource Estimation at Fog. *in In Consumer Communications and Networking Conference (CCNC),13th IEEE Annual*.

[2] Abdo, J. B., Demerjian, J., Chaouchi, H., and Atechian, T. (2015). Enhanced Revenue Optimizing SLA-Based Admission Control for IaaS Cloud Networks. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pages 225–230.

[3] Abdullah, R. (2013). Ontological Services Level Agreement (SLA) Model and Its Application in Cloud Computing Environment. *Conference on Software Engineering Parallel and Distributed Systems (SEPADS 13)*.

[4] Abrahão, S. and Insfran, E. (2017). Models@ Runtime for Monitoring Cloud Services in Google App Engine. In *2017 IEEE World Congress on Services (SERVICES)*, pages 30–35. IEEE.

[5] Abulkhair, M. F., Alkayal, E. S., and Jennings, N. R. (2017). Automated Negotiation Using Parallel Particle Swarm Optimization for Cloud Computing Applications. In *2017 International Conference on Computer and Applications (ICCA)*, pages 26–35. IEEE.

[6] Achtaich, A., Roudies, O., Souissi, N., Salinesi, C., and Mazo, R. (2019). Evaluation of the State-Constraint Transition Modelling Language: A Goal Question Metric Approach. In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B*, SPLC '19, page 106–113, New York, NY, USA. Association for Computing Machinery.

[7] Aghera, P., Chaudhary, S., and Kumar, V. (2012). An Approach to Build Multi-Tenant SaaS Application with Monitoring and SLA. In *Proceedings of the 2012 International Conference on Communication Systems and Network Technologies*, CSNT '12, pages 658–661, Washington, DC, USA. IEEE Computer Society.

[8] Ahuja, R., De, A., and Gabrani, G. (2011). SLA Based Scheduler for Cloud for Storage Computational Services. In *2011 International Conference on Computational Science and Its Applications*, pages 258–262.

[9] Ait-Idir, M. and Agoulmine, N. (2016). Enhancing Cloud Capabilities for SLA Enforcement of Cloud Scheduled Applications. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*, UCC '16, pages 298–303, New York, NY, USA. ACM.

[10] Ait-Idir, M., Cherkaoui, E. H., Rachkidi, E., Chendeb, N., and Agoulmine, N. (2014). MOSt-CB: SLA Enforcement and Smart VNE (Virtual Network Embedding) in a Multi Cloud Providers Environment. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 86–92. IEEE.

[11] Akbar, M., Sukmana, H. T., and Khairani, D. (2014). Models and Software Measurement Using Goal/Question/Metric Method and CMS Matrix Parameter (Case Study Discussion Forum). In *2014 International Conference on Cyber and IT Service Management (CITSM)*, pages 34–38.

[12] Al Etawi, N. A. (2018). A Comparison between Cluster, Grid, and Cloud Computing. *International Journal of Computer Applications*, 179(32):37–42.

[13] Al Falasi, A. and Serhani, M. A. (2016). SLA Specification and Negotiation Model for a Network of Federated Clouds: CloudLend. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 772–779. IEEE.

[14] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376.

[15] Al-Ghuwairi, A.-R., Khalaf, M. N., Al-Yasen, L., Salah, Z., Alsarhan, A., and Baarah, A. H. (2016). A Dynamic Model for Automatic Updating Cloud Computing SLA (DSLA). In *Proceedings of the International Conference on Internet of Things and Cloud Computing*, ICC '16, pages 57:1–57:7, New York, NY, USA. ACM.

[16] Al-Kiswany, S., , H., Liu, Z., and Sankaranarayanan, J. (2013). Cost Exploration of Data Sharings in the Cloud. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 601–612, New York, NY, USA. ACM.

[17] Al Muktadir, A. H., Jibiki, M., Martinez-Julia, P., and Kafle, V. P. (2018). Resource Negotiation Game for Cloud Networks with Limited Resources. In *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, pages 1–4. IEEE.

[18] Al-Shammari, S. and Al-Yasiri, A. (2015). MonSLAR: A Middleware for Monitoring SLA for Restful Services in Cloud Computing. In *2015 IEEE 9th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*, pages 46–50. IEEE.

[19] Al-Shara, Z., Alvares, F., Bruneliere, H., Lejeune, J., Prud'Homme, C., and Ledoux, T. (2018). Come4acloud: An End-to-End Framework for Autonomic Cloud Systems. *Future Generation Computer Systems*, 86:339–354.

[20] Alboghdady, S., Winter, S., Taha, A., Zhang, H., and Suri, N. (2017). C'Mon: Monitoring the Compliance of Cloud Services to Contracted Properties. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17, pages 36:1–36:6, New York, NY, USA. ACM.

[21] Alhamad, M., Dillon, T., and Chang, E. (2011). A Survey on SLA and Performance Measurement in Cloud Computing. In Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D. C., White, J., Hauswirth, M., Hitzler, P., and Mohania, M., editors, *On the Move to Meaningful Internet Systems: OTM 2011*, pages 469–477, Berlin, Heidelberg. Springer Berlin Heidelberg.

[22] Alhamad, M., Dillon, T., and Chang, E. (2011). Service Level Agreement for Distributed Services: A Review. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 1051–1054.

[23] Alhamazani, K., Wang, L., Rabhi, F., Mitra, K., and Ranjan, R. (2012). Cloud Monitoring for Optimizing the QoS of Hosted Applications. In *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, CLOUDCOM '12, pages 765–770, Washington, DC, USA. IEEE Computer Society.

[24] Alkandari, F. and Paige, R. F. (2012). Modelling and Comparing Cloud Computing Service Level Agreements. In *Proceedings of the 1st International Workshop on Model-Driven Engineering for High Performance and CLoud Computing*, MDHPCL '12, pages 3:1–3:6, New York, NY, USA. ACM.

[25] Alodib, M. (2014). Towards a Monitoring Framework for Cloud Services. In *Proceedings of the 2014 International Conference on Computational Science and Computational Intelligence - Volume 02*, CSCI '14, pages 146–151, Washington, DC, USA. IEEE Computer Society.

[26] Alodib, M. (2016). QoS-Aware Approach to Monitor Violations of SLAs in the IoT. *Journal of Innovation in Digital Ecosystems*, 3(2):197–207.

[27] Alqahtani, A., Li, Y., Patel, P., Solaiman, E., and Ranjan, R. (2018). End-to-End Service Level Agreement Specification for IoT Applications. In *2018 International Conference on High Performance Computing Simulation (HPCS)*, pages 926–935 .

[28] Alqahtani, A., Solaiman, E., Buyya, R., and Ranjan, Rajiv, P. (2016). End-to-End QoS Specification and Monitoring in the Internet of Things. *Newsletter, IEEE Technical Committee on Cybernetics for Cyber-Physical Systems*, 1(2).

[29] Alrokayan, M., Vahid Dastjerdi, A., and Buyya, R. (2014). SLA-Aware Provisioning and Scheduling of Cloud Resources for Big Data Analytics. In *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–8.

[30] Alsaffar, A. A., Pham, H. P., Hong, C.-S., Huh, E.-N., and Aazam, M. (2016). An Architecture of IoT Service Delegation and Resource Allocation Based on Collaboration between Fog and Cloud Computing. *Mobile Information Systems*, 2016.

[31] Alsrheed, F., El Rhalibi, A., Randles, M., and Merabti, M. (2014). Intelligent Agents for Automated Cloud Computing Negotiation. In *2014 international conference on Multimedia Computing and Systems (ICMCS)*, pages 1169–1174. IEEE.

[32] Alzahrani, E. J., Tari, Z., Zeephongsekul, P., Lee, Y. C., Alsadie, D., and Zomaya, A. Y. (2016). SLA-Aware Resource Scaling for Energy Efficiency. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 852–859.

[33] Alzubaidi, A., Solaiman, E., Patel, P., and Mitra, K. (2019). Blockchain-Based SLA Management in the Context of IoT. *IT Professional*.

[34] Amato, A., Di Martino, B., and Venticinque, S. (2012). Evaluation and Brokering of Service Level Agreements for Negotiation of Cloud Infrastructures. In *2012 International Conference for Internet Technology and Secured Transactions*, pages 144–149. IEEE.

[35] Amyot, D., Shamsaei, A., Kealey, J., Tremblay, E., Miga, A., Mussbacher, G., Alhaj, M., Tawhid, R., Braun, E., and Cartwright, N. (2012). Towards Advanced Goal Model Analysis with JUCMNav. In Castano, S., Vassiliadis, P., Lakshmanan, L. V., and Lee, M. L., editors, *Advances in Conceptual Modeling*, pages 201–210, Berlin, Heidelberg. Springer Berlin Heidelberg.

[36] Anastasi, G. F., Coppola, M., Dazzi, P., and Distefano, M. (2016). QoS Guarantees for Network Bandwidth in Private Clouds. *Procedia Computer Science*, 97:4–13.

[37] Andreolini, M., Colajanni, M., and Pietri, M. (2012). A Scalable Architecture for Real-Time Monitoring of Large Information Systems. In *Proceedings of the 2012 Second Symposium on Network Cloud Computing and Applications*, NCCA '12, pages 143–150, Washington, DC, USA. IEEE Computer Society.

[38] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M, P. (2007). Web Services Agreement Specification (WS-Agreement). *InOpen Grid Forum*, 128(1):216.

[39] Antonescu, A., Gomes, A., Robinson, P., and Braun, T. (2013a). SLA-Driven Predictive Orchestration for Distributed Cloud-Based Mobile Services. In *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 738–743.

[40] Antonescu, A., Oprescu, A., Demchenko, Y., d. Laat, C., and Braun, T. (2013b). Dynamic Optimization of SLA-Based Services Scaling Rules. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 282–289.

[41] Antonescu, A.-F. and Braun, T. (2014). Improving Management of Distributed Services Using Correlations and Predictions in SLA-Driven Cloud Computing Systems. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8. IEEE.

[42] Antonescu, A.-F. and Braun, T. (2016). Simulation of SLA-Based VM-Scaling Algorithms for Cloud-Distributed Applications. *Future Generation Computer Systems*, 54:260 – 273.

[43] Antonescu, A.-F., Robinson, P., and Braun, T. (2013). Dynamic SLA Management with Forecasting Using Multi-Objective Optimization. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 457–463. IEEE.

[44] Apat, H. K., Sahoo, B., and Maiti, P. (2018). Service Placement in Fog Computing Environment. In *2018 International Conference on Information Technology (ICIT)*, pages 272–277.

[45] Arianyan, E., Taheri, H., and Khoshdel, V. (2017). Novel Fuzzy Multi Objective DVFS-Aware Consolidation Heuristics for Energy and SLA Efficient Resource Management in Cloud Data Centers. *Journal of Network and Computer Applications*, 78:43 – 61.

[46] Ashouraei, M., Khezr, S. N., Benlamri, R., and Navimipour, N. J. (2018). A New SLA-Aware Load Balancing Method in the Cloud Using an Improved Parallel Task Scheduling Algorithm. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 71–76.

[47] Asin, A. and Gascon, D. (2012). 50 Sensor Applications for a Smarter World. *Libelium Comunicaciones Distribuidas, Tech. Rep*.

[48] Aslanpour, M. S. and Dashti, S. E. (2016). SLA-Aware Resource Allocation for Application Service Providers in the Cloud. In *2016 Second International Conference on Web Research (ICWR)*, pages 31–42.

[49] Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things: A Survey. *Computer Networks*, 54(15):2787 – 2805.

[50] Aversa, R., Panza, N., and Tasquier, L. (2015). An Agent-Based Platform for Cloud Applications Performance Monitoring. In *Proceedings of the 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, CISIS '15, pages 535–540, Washington, DC, USA. IEEE Computer Society.

[51] Aversa, R. and Tasquier, L. (2016). Design of an Agent Based Monitoring Framework for Federated Clouds. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 115–120. IEEE.

[52] Aversa, R., Tasquier, L., and Venticinque, S. (2013). Agents Based Monitoring of Heterogeneous Cloud Infrastructures. In *Proceedings of the 2013 IEEE 10th International Conference on Ubiquitous Intelligence & Computing and 2013 IEEE 10th International Conference on Autonomic & Trusted Computing*, UIC-ATC '13, pages 527–532, Washington, DC, USA. IEEE Computer Society.

[53] AWS, A. W. S. (2019). Amazon Kinesis Data Firehose FAQs. https://aws.amazon.com/kinesis/data-firehose/faqs/. (Accessed on 07/03/2019).

[54] Badidi, E. (2013). A Cloud Service Broker for SLA-Based SaaS Provisioning. In *International Conference on Information Society (i-Society 2013)*, pages 61–66.

[55] Bahga, A. and Madisetti, V. K. (2016). Blockchain Platform for Industrial Internet of Things. *Journal of Software Engineering and Applications*, 9(10):533.

[56] Bai, X., Li, M., Huang, X., Tsai, W.-T., and Gao, J. (2013). Vee@Cloud: The Virtual Test Lab on the Cloud. In *Proceedings of the 8th International Workshop on Automation of Software Test*, AST '13, pages 15–18, Piscataway, NJ, USA. IEEE Press.

[57] Baig, R., Khan, W. A., Haq, I. U., and Khan, I. M. (2017). Agent-Based SLA Negotiation Protocol for Cloud Computing. In *2017 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, pages 33–37. IEEE.

[58] Bakraouy, Z., Baina, A., and Bellafkih, M. (2018). Autonomous SLAs Negotiation Based on Agreement-Broker: Services Availability. In *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, pages 48–53. IEEE.

[59] Balagoni, Y. and Rao, R. R. (2016). A Cost-Effective SLA-Aware Scheduling for Hybrid Cloud Environment. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–7.

[60] Banaie, F., Misic, J., Misic, V. B., Yaghmaee, M. H., and Hosseini, S. A. (2018). Performance Analysis of Multithreaded IoT Gateway. *IEEE Internet of Things Journal*.

[61] Banavar, G., Chandra, T., Mukherjee, B., Nagarajarao, J., Strom, R. E., and Sturman, D. C. (1999). An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*, pages 262–272. IEEE.

[62] Bar, P., Benfredj, R., Marks, J., Ulevinov, D., Wozniak, B., Casale, G., and Knottenbelt, W. J. (2013). Towards a Monitoring Feedback Loop for Cloud Applications. In *Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds*, MultiCloud '13, pages 43–44, New York, NY, USA. ACM.

[63] Bartoletti, M. and Pompianu, L. (2017). An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns. In Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P. Y., Teague, V., Bracciali, A., Sala, M., Pintore, F., and Jakobsson, M., editors, *Financial Cryptography and Data Security*, pages 494–509, Cham. Springer International Publishing.

[64] Basili, V. (1992). Software Modeling and Measurement: The Goal/Question/Metric Paradigm. University of Maryland. Technical report, CS-TR-2956, UMIACS-TR-92-96.

[65] Basili, V. R. and Rombach, H. D. (1988). The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, 14(6):758–773.

[66] Basili, V. R. and Selby, R. W. (1984). Data Collection and Analysis in Software Research and Management. *Proceedings of the American Statistical Association and Biomeasure Society*, pages 13–16.

[67] Basili, V. R. and Weiss, D. M. (1984). A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, SE-10(6):728–738.

[68] Belk, M., Papatheocharous, E., Germanakos, P., and Samaras, G. (2012). Investigating the Relation between Users' Cognitive Style and Web Navigation Behavior with K-Means Clustering. In Castano, S., Vassiliadis, P., Lakshmanan, L. V., and Lee, M. L., editors, *Advances in Conceptual Modeling*, pages 337–346, Berlin, Heidelberg. Springer Berlin Heidelberg.

[69] Ben Jemaa, F., Pujolle, G., and Pariente, M. (2016). QoS-Aware VNF Placement Optimization in Edge-Central Carrier Cloud Architecture. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7.

[70] Benali, R., Teyeb, H., Balma, A., Tata, S., and Ben Hadj-Alouane, N. (2016). Evaluation of Traffic-Aware VM Placement Policies in Distributed Cloud Using CloudSim. In *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 95–100.

[71] Bendriss, J., Yahia, I. G. B., Riggio, R., and Zeghlache, D. (2018). A Deep Learning Based SLA Management for NFV-Based Services. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–3. IEEE.

[72] Benedictis, A. d., Rak, M., Turtur, M., and Villano, U. (2015). REST-Based SLA Management for Cloud Applications. In *Proceedings of the 2015 IEEE 24th International Conference on Enabling Technologies: Infrastructure for*

Collaborative Enterprises, WETICE '15, pages 93–98, Washington, DC, USA. IEEE Computer Society.

[73] Bermudez, I., Traverso, S., Munafò, M., and Mellia, M. (2014). A Distributed Architecture for the Monitoring of Clouds and CDNs: Applications to Amazon AWS. *IEEE Transactions on Network and Service Management*, 11(4):516–529.

[74] Bertolino, A., Calabrò, A., and De Angelis, G. (2013). Adaptive SLA Monitoring of Service Choreographies Enacted on the Cloud. In *2013 IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*, pages 92–101. IEEE.

[75] Bhargava, B., Angin, P., Ranchal, R., and Lingayat, S. (2015). A Distributed Monitoring and Reconfiguration Approach for Adaptive Network Computing. In *Proceedings of the 2015 IEEE 34th Symposium on Reliable Distributed Systems Workshop (SRDSW)*, SRDSW '15, pages 31–35, Washington, DC, USA. IEEE Computer Society.

[76] Bhuyan, B., Sarma, H. K. D., Sarma, N., Kar, A., and Mall, R. (2010). Quality of Service (QoS) Provisions in Wireless Sensor Networks and Related Challenges. *Wireless Sensor Network*, 2(11):861.

[77] Bianco, P., Lewis, G., and Merson, Paulo, P. (2008). Service Level Agreements in Service-Oriented Architecture Environments. *Technical Report CMU/SEI-2008-TN-021, Carnegie Mellon*.

[78] Binu, V. and Gangadhar, N. (2014). A Cloud Computing Service Level Agreement Framework with Negotiation and Secure Monitoring. In *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–8. IEEE.

[79] Biswas, M. I., Parr, G., McClean, S., Morrow, P., and Scotney, B. (2014). SLA-Based Scheduling of Applications for Geographically Secluded Clouds. In *2014 International Conference and Workshop on the Network of the Future (NOF)*, volume Workshop, pages 1–8.

[80] Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., and Parashar, M. (2017). Mobility-aware Application Scheduling in Fog Computing. *IEEE Cloud Computing*, 4(2):26–35.

[81] Bochicchio, M. A., Longo, A., and Secco, A. (2013). An Online Laboratory for SLA Management. In *2013 IEEE Global Engineering Education Conference (EDUCON)*, pages 1130–1136. IEEE.

[82] Boehm, B. W., Brown, J. R., and Lipow, M. (1976). Quantitative Evaluation of Software Quality. In *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE '76, pages 592–605, Los Alamitos, CA, USA. IEEE Computer Society Press.

[83] Boloor, K., Chirkova, R., Viniotis, Y., and Salo, T. (2010). Dynamic Request Allocation and Scheduling for Context Aware Applications Subject to a Percentile Response Time SLA in a Distributed Cloud. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 464–472.

[84] Boniface, M., Nasser, B., Papay, J., Phillips, S. C., Servin, A., Yang, X., Zlatev, Z., Gogouvitis, S. V., Katsaros, G., Konstanteli, K., Kousiouris, G., Menychtas, A., and Kyriazis, D. (2010). Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds. In *Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services*, ICIW '10, pages 155–160, Washington, DC, USA. IEEE Computer Society.

[85] Booth, A., Alqahtani, A., and Solaiman, E. (2020). IoT Monitoring with Blockchain: Generating Smart Contracts from Service Level Agreements. In *Managing Internet of Things Applications Across Edge and Cloud Datacenters*. IET.

[86] Borges, H. P., de Souza, J. N., Schulze, B., and Mury, A. R. (2014). Automatic Services Instantiation Based on a Process Specification. *J. Netw. Comput. Appl.*, 39(C):1–16.

[87] Borgetto, D., Maurer, M., Da-Costa, G., Pierson, J., and Brandic, I. (2012). Energy-Efficient and SLA-Aware Management of IaaS Clouds. In *2012 Third International Conference on Future Systems: Where Energy, Computing and Communication Meet (e-Energy)*, pages 1–10.

[88] Borgetto, D., Maurer, M., Da-Costa, G., Pierson, J.-M., and Brandic, I. (2012). Energy-Efficient and SLA-Aware Management of IaaS Clouds. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12, pages 25:1–25:10, New York, NY, USA. ACM.

[89] Bouchenak, S. (2010). Automated Control for SLA-Aware Elastic Clouds. In *Proceedings of the Fifth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*, pages 27–28.

[90] Brandic, I., Music, D., and Dustdar, S. (2009). Service Mediation and Negotiation Bootstrapping As First Achievements Towards Self-Adaptable Grid and Cloud Services. In *Proceedings of the 6th International Conference Industry Session on Grids Meets Autonomic Computing*, GMAC '09, pages 1–8, New York, NY, USA. ACM.

[91] Brandic, I., Music, D., Dustdar, S., Venugopal, S., and Buyya, R. (2008). Advanced QoS Methods for Grid Workflows Based on Meta-Negotiations and SLA-Mappings. In *2008 Third Workshop on Workflows in Support of Large-Scale Science*, pages 1–10. IEEE.

[92] Breitgand, D., Dubitzky, Z., Epstein, A., Glikson, A., and Shapira, I. (2012). SLA-Aware Resource Over-Commit in an IaaS Cloud. In *2012 8th international*

conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm), pages 73–81.

[93] Breitgand, D. and Epstein, A. (2011). SLA-Aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 161–168.

[94] Brogi, A., Forti, S., Guerrero, C., and Lera, I. (2019). Meet Genetic Algorithms in Monte Carlo: Optimised Placement of Multi-Service Applications in the Fog. In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 13–17.

[95] Buyya, R. (2009). Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 1–1.

[96] Buyya, R. and Dastjerdi, A. V. (2016). *Internet of Things: Principles and Paradigms*. Elsevier.

[97] Calbimonte, J.-P., Riahi, M., Kefalakis, N., Soldatos, J., and Zaslavsky, A. (2014). Utility Metrics Specifications. OpenIoT Deliverable D422. Technical report, Infoscience, the École Polytechnique Fédérale de Lausanne (EPFL).

[98] Caldiera, V. R. B. G. and Rombach, H. D. (1994). The Goal Question Metric Approach. *Encyclopedia of software engineering*, pages 528–532.

[99] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.

[100] Cao, Z. and Dong, S. (2012). Dynamic VM Consolidation for Energy-Aware and SLA Violation Reduction in Cloud Computing. In *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 363–369.

[101] Cardellini, V., Casalicchio, E., Lo Presti, F., and Silvestri, L. (2011). SLA-Aware Resource Management for Application Service Providers in the Cloud. In *2011 First International Symposium on Network Cloud Computing and Applications*, pages 20–27.

[102] Casola, V., Benedictis, A. D., and Rak, M. (2015). Security Monitoring in the Cloud: An SLA-Based Approach. In *Proceedings of the 2015 10th International Conference on Availability, Reliability and Security*, ARES '15, pages 749–755, Washington, DC, USA. IEEE Computer Society.

[103] Cedillo, P., Jimenez-Gomez, J., Abrahao, S., and Insfran, E. (2015). Towards a Monitoring Middleware for Cloud Services. In *Proceedings of the 2015 IEEE International Conference on Services Computing*, SCC '15, pages 451–458, Washington, DC, USA. IEEE Computer Society.

[104] Chai, A., Bazm, M., Camarasu-Pop, S., Glatard, T., Benoit-Cattin, H., and Suter, F. (2017). Modeling Distributed Platforms from Application Traces for Realistic File Transfer Simulation. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 54–63.

[105] Chang, C., Lai, K., and Yang, C. (2013). Auction-Based Resource Provisioning with SLA Consideration on Multi-Cloud Systems. In *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, pages 445–450.

[106] Chaqfeh, M. A. and Mohamed, N. (2012). Challenges in Middleware Solutions for the Internet of Things. In *2012 International Conference on Collaboration Technologies and Systems (CTS)*, pages 21–26.

[107] Chen, J., Tsai, C., Lu, S., Luc, S., and Abedin, F. (2015). Resource Reallocation Based on SLA Requirement in Cloud Environment. In *2015 IEEE 12th International Conference on e-Business Engineering*, pages 377–381.

[108] Chen, M., Mao, S., and Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, 19(2):171–209.

[109] Cheng, S., Cao, C., Yu, P., and Ma, X. (2016). SLA-Aware and Green Resource Management of IaaS Clouds. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 457–464.

[110] Chhetri, M. B., Vo, Q. B., and Kowalczyk, R. (2014). Adaptive AutoSLAM - Policy-Based Orchestration of SLA Establishment. In *2014 IEEE International Conference on Services Computing*, pages 472–479.

[111] Chhetri, M. B., Vo, Q. B., and Kowalczyk, R. (2016). CL-SLAM: Cross-Layer SLA Monitoring Framework for Cloud Service-Based Applications. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*, UCC '16, pages 30–36, New York, NY, USA. ACM.

[112] Chi, Y., Moon, H. J., and Hacigümüş, H. (2011). ICBS: Incremental Cost-Based Scheduling Under Piecewise Linear SLAs. *Proc. VLDB Endow.*, 4(9):563–574.

[113] Chituc, C.-M. (2015). Towards a Methodology for Trade-off Analysis in a Multi-Cloud Environment Considering Monitored QoS Metrics and Economic Performance Assessment Results. In *Proceedings of the 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, CLOUDCOM '15, pages 479–482, Washington, DC, USA. IEEE Computer Society.

[114] Chokhani, P. and Somani, G. (2013). Dynamic Resource Allocation Using Auto-Negotiation in Haizea. In *2013 Sixth International Conference on Contemporary Computing (IC3)*, pages 232–238. IEEE.

[115] Chung-Cheng Li and Kuochen Wang (2014). An SLA-Aware Load Balancing Scheme for Cloud Datacenters. In *The International Conference on Information Networking 2014 (ICOIN2014)*, pages 58–63.

[116] Cicotti, G., D'Antonio, S., Cristaldi, R., and Sergio, A. (2013). How to Monitor QoS in Cloud Infrastructures: The QoSMONaaS Approach. In Fortino, G., Badica, C., Malgeri, M., and Unland, R., editors, *Intelligent Distributed Computing VI*, pages 253–262, Berlin, Heidelberg. Springer Berlin Heidelberg.

[117] Cirani, S., Ferrari, G., Iotti, N., and Picone, M. (2015). The iot hub: A fog node for seamless management of heterogeneous connected smart objects. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops)*, pages 1–6. IEEE.

[118] Cloud, H. (2011). The NIST Definition of Cloud Computing. *National Institute of Science and Technology, Special Publication, 800*, 145.

[119] Comuzzi, M. and Pernici, B. (2009). A Framework for QoS-Based Web Service Contracting. *ACM Transactions on the Web (TWEB)*, 3(3):10.

[120] Copil, G., Moldovan, D., Salomie, I., Cioara, T., Anghel, I., and Borza, D. (2012). Cloud SLA Negotiation for Energy Saving—a Particle Swarm Optimization Approach. In *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*, pages 289–296. IEEE.

[121] Council, C. S. C. (2011). Practical Guide to Cloud Service Agreements Version 2.0. Technical Report Supplement C, The Object Management Group (OMG).

[122] Crecana, C.-C. and Pop, F. (2018). Monitoring-Based Auto-Scalability Across Hybrid Clouds. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, pages 1087–1094, New York, NY, USA. ACM.

[123] da Cunha Rodrigues, G., Calheiros, R. N., dos Santos, G. L., Guimaraes, V. T., Granville, L. Z., Tarouco, L., and Buyya, R. (2018). Unfolding the Mutual Relation between Timeliness and Scalability in Cloud Monitoring. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00772–00778. IEEE.

[124] da Silva, T. L. C., Nascimento, M. A., Macêdo, J. A. F., Sousa, F. R. C., and Machado, J. C. (2012). Towards Non-Intrusive Elastic Query Processing in the Cloud. In *Proceedings of the Fourth International Workshop on Cloud Data Management*, CloudDB '12, pages 9–16, New York, NY, USA. ACM.

[125] D'Angelo, G., Ferretti, S., and Marzolla, M. (2018). A Blockchain-Based Flight Data Recorder for Cloud Accountability. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, CryBlock'18, pages 93–98, New York, NY, USA. ACM.

[126] Daniel, D. and Lovesum, S. J. (2011). A Novel Approach for Scheduling Service Request in Cloud with Trust Monitor. In *2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies*, pages 509–513. IEEE.

[127] Darweesh, S. A., Ebrahim, G. A., and Bedour, H. M. S. (2019). Evaluating Multi-Agent System Security Using Goal/Question/Metric Approach and Fuzzy Logic. In *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 1–6.

[128] Das, K. and Behera, R. N. (2017). A Survey on Machine Learning: Concept, Algorithms and Applications. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(2):1301–1309.

[129] Dastjerdi, A. V. and Buyya, R. (2012). An Autonomous Reliability-Aware Negotiation Strategy for Cloud Computing Environments. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, CCGRID '12, pages 284–291, Washington, DC, USA. IEEE Computer Society.

[130] Dastjerdi, A. V. and Buyya, R. (2015). An Autonomous Time-Dependent SLA Negotiation Strategy for Cloud Computing. *The Computer Journal*, 58(11):3202–3216.

[131] Dementyev, A., Hodges, S., Taylor, S., and Smith, J. (2013). Power Consumption Analysis of Bluetooth Low Energy, ZigBee and ANT Sensor Nodes in a Cyclic Sleep Scenario. In *2013 IEEE International Wireless Symposium (IWS)*, pages 1–4.

[132] Derrick, B. and White, P. (2017). Comparing two samples from an individual Likert question. *International Journal of Mathematics and Statistics*, 18.

[133] Dey, S. (2018). A Secure Fair Queuing and SLA Based Slotted Round Robin Load Balancing Approach for Cloud Data Centers. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 382–387.

[134] Dhingra, M., Lakshmi, J., and Nandy, S. K. (2012). Resource Usage Monitoring in Clouds. In *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*, GRID '12, pages 184–191, Washington, DC, USA. IEEE Computer Society.

[135] Díaz, M., Martín, C., and Rubio, B. (2016). State-of-the-Art, Challenges, and Open Issues in the Integration of Internet of Things and Cloud Computing. *Journal of Network and Computer Applications*, 67:99–117.

[136] Ding, J. and Zhao, Z. (2012). Towards Autonomic SLA Management: A Review. In *2012 International Conference on Systems and Informatics (ICSAI2012)*, pages 2552–2555. IEEE.

[137] Dingle, N. J., Knottenbelt, W. J., and Wang, L. (2008). Service Level Agreement Specification, Compliance Prediction and Monitoring with Performance Trees. In *22nd Annual European Simulation and Modelling Conference (ESM 2008)*, pages 137–14.

[138] Distefano, S., Puliafito, A., Rak, M., Venticinque, S., Villano, U., Cuomo, A., Di Modica, G., and Tomarchio, O. (2011). QoS Management in Cloud@Home Infrastructures. In *Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, CYBERC '11, pages 190–197, Washington, DC, USA. IEEE Computer Society.

[139] Djemai, T., Stolf, P., Monteil, T., and Pierson, J. (2019). A Discrete Particle Swarm Optimization Approach for Energy-Efficient IoT Services Placement Over Fog Infrastructures. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 32–40.

[140] Dsouza, C., Ahn, G.-J., and Taguinod, Marthony, P. (2014). Policy-Driven Security Management for Fog Computing: Preliminary Framework and a Case Study. *in on IEEE 15th International Conference In Information Reuse and Integration (IRI)*.

[141] Duan, R., Chen, X., and Xing, T. (2011). A QoS Architecture for IoT. In *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pages 717–720.

[142] Edu-yaw, T. and Kuada, E. (2018). Service Level Agreement Negotiation and Monitoring System in Cloud Computing. In *2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST)*, pages 1–8. IEEE.

[143] El Kafhali, S. and Salah, K. (2017). Efficient and Dynamic Scaling of Fog Nodes for IoT Devices. *The Journal of Supercomputing*, 73(12):5261–5284.

[144] El Kafhali, S. and Salah, K. (2018). Performance Modelling and Analysis of Internet of Things Enabled Healthcare Monitoring Systems. *IET Networks*, 8(1):48–58.

[145] El-Matary, D., El-Attar, N., Awad, W., and Hanafy, I. (2019). Automated Negotiation Framework Based on Intelligent Agents for Cloud Computing. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 156–161. IEEE.

[146] El-Sayed, H., Sankar, S., Prasad, M., Puthal, D., Gupta, A., Mohanty, M., and Lin, C. (2018). Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment. *IEEE Access*, 6:1706–1717.

[147] Eldawy, A. and Mokbel, M. F. (2015). Spatialhadoop: A Mapreduce Framework for Spatial Data. In *2015 IEEE 31st international conference on Data Engineering*, pages 1352–1363. IEEE.

[148] Elhadi, S., Marzak, A., Sael, N., and Merzouk, S. (2018). Comparative Study of IoT Protocols. *Smart Application and Data Analysis for Smart Cities (SADASC'18)*.

[149] Elliott, K., Massacci, F., Ngo, C.-N., and Williams, J. M. (2016). Unruly Innovation: Distributed Ledgers, Blockchains and the Protection of Transactional Rents. *Blockchains and the Protection of Transactional Rents (December 22, 2016)*.

[150] Emeakaroha, V. C., Brandic, I., Maurer, M., and Breskovic, I. (2011). SLA-Aware Application Deployment and Resource Allocation in Clouds. In *2011 IEEE 35th Annual Computer Software and Applications Conference Workshops*, pages 298–303.

[151] Emeakaroha, V. C., Brandic, I., Maurer, M., and Dustdar, S. (2010). Low Level Metrics to High Level SLAs - LoM2HiS Framework: Bridging the Gap between Monitored Metrics and SLA Parameters in Cloud Environments. In *2010 International Conference on High Performance Computing Simulation*, pages 48–54.

[152] Emeakaroha, V. C., Ferreto, T. C., Netto, M. A. S., Brandic, I., and De Rose, C. A. F. (2012). CASViD: Application Level Monitoring for SLA Violation Detection in Clouds. In *Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference*, COMPSAC '12, pages 499–508, Washington, DC, USA. IEEE Computer Society.

[153] Engel, R., Rajamoni, S., Chen, B., Ludwig, H., and Keller, A. (2018). YSLA: Reusable and Configurable SLAs for Large-Scale SLA Management. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 317–325. IEEE.

[154] Fabric., H. (2019). Building Your First Network — Hyperledger-Fabricdocs Master Documentation. https://hyperledger-fabric.readthedocs.io/en/release-1.4/build_network.html. (Accessed on 09/29/2019).

[155] Falasi, A. A., Serhani, M. A., and Dssouli, R. (2013). A Model for Multi-Levels SLA Monitoring in Federated Cloud Environment. In *Proceedings of the 2013 IEEE 10th International Conference on Ubiquitous Intelligence & Computing and 2013 IEEE 10th International Conference on Autonomic & Trusted Computing*, UIC-ATC '13, pages 363–370, Washington, DC, USA. IEEE Computer Society.

[156] Faniyi, F. and Bahsoon, R. (2011). Engineering Proprioception in SLA Management for Cloud Architectures. In *Proceedings of the 2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, WICSA '11, pages 336–340, Washington, DC, USA. IEEE Computer Society.

[157] Faniyi, F. and Bahsoon, R. (2012). Self-Managing SLA Compliance in Cloud Architectures: A Market-Based Approach. In *Proceedings of the 3rd International ACM SIGSOFT Symposium on Architecting Critical Systems*, ISARCS '12, pages 61–70, New York, NY, USA. ACM.

[158] Faniyi, F. and Bahsoon, R. (2015). A Systematic Review of Service Level Management in the Cloud. *ACM Comput. Surv.*, 48(3):43:1–43:27.

[159] Farokhi, S. (2014). Towards an SLA-Based Service Allocation in Multi-Cloud Environments. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 591–594.

[160] Flammini, A. and Sisinni, E. (2014). Wireless Sensor Networking in the Internet of Things and Cloud Computing Era. *Procedia Engineering*, 87:672 – 679.

[161] Foster, H. and Spanoudakis, G. (2011a). Advanced Service Monitoring Configurations with SLA Decomposition and Selection. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1582–1589. ACM.

[162] Foster, H. and Spanoudakis, G. (2011b). SMaRT: A Workbench for Reporting the Monitorability of Services from SLAs. In *Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems*, pages 36–42. ACM.

[163] Freitas, A. L., Parlavantzas, N., and Pazat, J.-L. (2010). A QoS Assurance Framework for Distributed Infrastructures. In *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond*, MONA '10, pages 1–8, New York, NY, USA. ACM.

[164] Furtado, T., Francesquini, E., Lago, N., and Kon, F. (2014). A Middleware for Reflective Web Service Choreographies on the Cloud. In *Proceedings of the 13th Workshop on Adaptive and Reflective Middleware*, ARM '14, pages 9:1–9:6, New York, NY, USA. ACM.

[165] Gaillard, G., Barthel, D., Theoleyre, F., and Valois, F. (2014). *SLA Specification for IoT Operation-The WSN-SLA Framework*. PhD thesis, INRIA.

[166] Galati, A., Djemame, K., Fletcher, M., Jessop, M., Weeks, M., and McAvoy, J. (2014). A WS-Agreement Based SLA Implementation for the CMAC Platform. In Altmann, J., Vanmechelen, K., and Rana, O. F., editors, *Economics of Grids, Clouds, Systems, and Services*, pages 159–171, Cham. Springer International Publishing.

[167] Gámez Díaz, A., Fernández Montes, P., and Ruiz Cortés, A. (2018). Fostering SLA-Driven API Specifications. *JCIS 2018: XIV Jornadas de Ciencia e Ingeniería de Servicios (2018),*.

[168] Gantz, J. and Reinsel, D. (2012). The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. *IDC iView: IDC Analyze the future*, 2007(2012):1–16.

[169] García García, A., Blanquer Espert, I., and Hernández García, V. (2014). SLA-Driven Dynamic Cloud Resource Management. *Future Gener. Comput. Syst.*, 31:1–11.

[170] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., and Riviere, E. (2015). Edge-Centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev.*, 45(5):37–42.

[171] Garg, S. K., Toosi, A. N., Gopalaiyengar, S. K., and Buyya, R. (2014). SLA-Based Virtual Machine Management for Heterogeneous Workloads in a Cloud Datacenter. *Journal of Network and Computer Applications*, 45:108 – 120.

[172] Ghosh, N. and Ghosh, S. K. (2012). An Approach to Identify and Monitor SLA Parameters for Storage-as-a-Service Cloud Delivery Model. In *2012 IEEE Globecom Workshops*, pages 724–729. IEEE.

[173] Ghumman, W. A. and Schill, A. (2017). Continuous and Distributed Monitoring of Cloud SLAs Using S3LACC. In *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 114–119. IEEE.

[174] Ghumman, W. A., Schill, A., and Lässig, J. (2016). The Flip-Flop SLA Negotiation Strategy Using Concession Extrapolation and 3D Utility Function. In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, pages 159–168. IEEE.

[175] Gomes, R. L., Bittencourt, L. F., and Madeira, E. R. (2012). A Generic SLA Negotiation Protocol for Virtualized Environments. In *2012 18th IEEE International Conference on Networks (ICON)*, pages 7–12. IEEE.

[176] Gope, P. and Hwang, T. (2015). BSN-Care: A Secure IoT-Based Modern Healthcare System Using Body Sensor Network. *IEEE sensors journal*, 16(5):1368–1376.

[177] Goudarzi, H. and Pedram, M. (2011). Multi-Dimensional SLA-Based Resource Allocation for Multi-Tier Cloud Computing Systems. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 324–331.

[178] Goudarzi, H. and Pedram, M. (2016). Hierarchical SLA-Driven Resource Management for Peak Power-Aware and Energy-Efficient Operation of a Cloud Datacenter. *IEEE Transactions on Cloud Computing*, 4(2):222–236.

[179] Grady, R. B. and Caswell, D. L. (1987). *Software Metrics: Establishing a Company-Wide Program*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[180] Grati, R., Boukadi, K., and Ben-Abdallah, H. (2014). A Framework for IaaS-to-SaaS monitoring of BPEL Processes in the Cloud: Design and Evaluation. In *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, pages 557–564. IEEE.

[181] Groléat, T. and Pouyllau, H. (2011). Distributed Inter-Domain SLA Negotiation Using Reinforcement Learning. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 33–40. IEEE.

[182] Groleat, T. and Pouyllau, H. (2012). Distributed Learning Algorithms for Inter-NSP SLA Negotiation Management. *IEEE Transactions on Network and Service Management*, 9(4):433–445.

[183] Gu, L., Zeng, D., Guo, S., Barnawi, A., and Xiang, Yong, P. (2015). Cost-Efficient Resource Management in Fog Computing Supported Medical CPS. *in IEEE Transactions on Emerging Topics in Computing*.

[184] Gunter, D., Tierney, B., Crowley, B., Holding, M., and Lee, J. (2000). NetLogger: A Toolkit for Distributed System Performance Analysis. In *Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (Cat. No.PR00728)*, pages 267–273.

[185] Guo, C., Yuan, L., Xiang, D., Dang, Y., Huang, R., Maltz, D., Liu, Z., Wang, V., Pang, B., Chen, H., Lin, Z.-W., and Kurien, V. (2015). Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 139–152, New York, NY, USA. ACM.

[186] Gupta, H., Dastjerdi, A. V., Ghosh, S. K., and Buyya, R. (2017). IFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments. *Software: Practice and Experience*, 47(9):1275–1296.

[187] Gutierrez-Garcia, J. O. and Sim, K. M. (2010). Self-Organizing Agents for Service Composition in Cloud Computing. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 59–66.

[188] Hail, M. A. M., Amadeo, M., Molinaro, A., and Fischer, S. (2015). On the Performance of Caching and Forwarding in Information-Centric Networking for the IoT. In *International Conference on Wired/Wireless Internet Communication*, pages 313–326. Springer.

[189] Haiteng, Z., Zhiqing, S., Hong, Z., and Jie, Z. (2012). Establishing Service Level Agreement Requirement Based on Monitoring. In *Proceedings of the 2012 Second International Conference on Cloud and Green Computing*, CGC '12, pages 472–476, Washington, DC, USA. IEEE Computer Society.

[190] Halabi, T. and Bellaiche, M. (2018). A Broker-Based Framework for Standardization and Management of Cloud Security-SLAs. *Computers & Security*, 75:59–71.

[191] Halili, M. K. and Çiço, B. (2018). Towards Custom Tailored SLA in IaaS Environment through Negotiation Model: An Overview. In *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4. IEEE.

[192] Hammadi, A., Hussain, O. K., Dillon, T., and Hussain, F. K. (2013). A Framework for SLA Management in Cloud Computing for Informed Decision Making. *Cluster Computing*, 16(4):961–977.

[193] Hani, A. F. M., Paputungan, I. V., H, M. F., and A, V. S. (2017). Manifold Learning in SLA Violation Detection and Prediction for Cloud-Based System. In *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*, ICC '17, pages 183:1–183:5, New York, NY, USA. ACM.

[194] Hani, A. F. M., Paputungan, I. V., and Hassan, M. F. (2015). Renegotiation in Service Level Agreement Management for a Cloud-Based System. *ACM Comput. Surv.*, 47(3):51:1–51:21.

[195] Harel, D., Pnueli, A., Schmidt, J. P., and Sherman, R. (1987). On the Formal Semantics of Statecharts (Extended Abstract). In *Proceedings of the Second Annual IEEE Symposium on Logic in Computer Science (LICS 1987)*, pages 54–64. IEEE Computer Society Press.

[196] Hasan, M. S., Kouki, Y., Ledoux, T., and Pazat, J.-L. (2015). Exploiting Renewable Sources: When Green SLA Becomes A Possible Reality in Cloud Computing. *IEEE Transactions on Cloud Computing*, 5(2):249–262.

[197] Hashmi, K., Najmi, E., Malik, Z., Medjahed, B., Alhosban, A., and Rezgui, A. (2014). Automated Negotiation Using Semantic Rules. In *Proceedings of the 2014 IEEE International Conference on Services Computing*, SCC '14, pages 536–543, Washington, DC, USA. IEEE Computer Society.

[198] Hayat, R., Sabir, E., Badidi, E., and ElKoutbi, M. (2017). A Signaling Game-Based Approach for Data-as-a-Service Provisioning in IoT-Cloud. *Future Generation Computer Systems*.

[199] Hazarika, B., Singh, T. J., and Al Saedy, H. (2015). Window Based State Monitoring in Cloud Datacenter Using VM Concept. *Procedia Computer Science*, 70:618–624.

[200] He, H., Ma, Z., Chen, H., and Shao, W. (2013). Towards an SLA-Driven Cache Adjustment Approach for Applications on PaaS. In *Proceedings of the 5th Asia-Pacific Symposium on Internetware*, Internetware '13, pages 11:1–11:10, New York, NY, USA. ACM.

[201] He, H., Ma, Z., Chen, H., Yeh, C. Y., and Shao, W. (2014). An Aspect-Oriented Approach to SLA-Driven Monitoring Multi-tenant Cloud Application. In *Proceedings of the 2014 IEEE International Conference on Cloud Computing*, CLOUD '14, pages 857–864, Washington, DC, USA. IEEE Computer Society.

[202] He, X., Ren, Z., Shi, C., and Fang, J. (2016). A Novel Load Balancing Strategy of Software-Defined Cloud/Fog Networking in the Internet of Vehicles. *China Communications*, 13(Supplement2):140–149.

[203] Hlaing, H. H., Kanemitsu, H., Nakajima, T., and Nakazato, H. (2019). On the Optimal Number of Computational Resources in MapReduce. In *International Conference on Cloud Computing*, pages 240–252. Springer.

[204] Holloway, M., Schuller, D., and Steinmetz, R. (2015). Customized Cloud Service Quality: Approaching Pareto-Efficient Outcomes in Concurrent Multiple-Issue Negotiations. In *Proceedings of the 8th International Conference on Utility and Cloud Computing*, UCC '15, pages 256–260, Piscataway, NJ, USA. IEEE Press.

[205] Huang, C.-J., Guan, C.-T., Chen, H.-M., Wang, Y.-W., Chang, S.-C., Li, C.-Y., and Weng, C.-H. (2013). An Adaptive Resource Management Scheme in Cloud Computing. *Engineering Applications of Artificial Intelligence*, 26(1):382–389.

[206] Hughes, A., Park, A., Kietzmann, J., and Archer-Brown, C. (2019). Beyond Bitcoin: What blockchain and Distributed Ledger Technologies Mean for Firms. *Business Horizons*, 62.

[207] Hung, P. C., Li, H., and Jeng, J.-J. (2004). WS-Negotiation: An Overview of Research Issues. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pages 10–pp. IEEE.

[208] Hussain, W., Hussain, F. K., Hussain, O. K., Damiani, E., and Chang, E. (2017). Formulating and Managing Viable SLAs in Cloud Computing from a Small to Medium Service Provider's Viewpoint: A State-of-the-Art Review. *Information Systems*, 71:240–259.

[209] Hussain, W., Hussain, F. K., Saberi, M., Hussain, O. K., and Chang, E. (2018). Comparing Time Series with Machine Learning-Based Prediction Approaches for Violation Management in Cloud SLAs. *Future Generation Computer Systems*, 89:464–477.

[210] Hyperledger (2018). Hyperledger Fabric – Hyperledger. https://www.hyperledger.org/projects/fabric. (Accessed on 09/11/2019).

[211] Ibrahim, A. A. Z. A. (2018). PRESEnCE: A Framework for Monitoring, Modelling and Evaluating the Performance of Cloud SaaS Web Services. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 83–86. IEEE.

[212] Intharawijitr, K., Iida, K., and Koga, Hiroyuki, P. (2016). Analysis of Fog Model Considering Computing and Communication Latency in 5G Cellular Networks I. *in In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*.

[213] Jacob, B., Brown, M., Fukui, K., Trivedi, N., et al. (2005). Introduction to Grid Computing. *IBM redbooks*, pages 3–6.

[214] Jayaraman, P. P., Mitra, K., Saguna, S., Shah, T., Georgakopoulos, D., and Ranjan, Rajiv, P. (2015). Orchestrating Quality of Service in the Cloud of Things Ecosystem. *In 2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pages 185–190.

[215] Jayathilaka, H., Krintz, C., and Wolski, R. (2015). Response Time Service Level Agreements for Cloud-Hosted Web Applications. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, SoCC '15, pages 315–328, New York, NY, USA. ACM.

[216] Jie, Y., Tang, X., Choo, K.-K. R., Su, S., Li, M., and Guo, C. (2018). Online Task Scheduling for Edge Computing Based on Repeated Stackelberg Game. *Journal of Parallel and Distributed Computing*, 122:159 – 172.

[217] Kaaniche, N., Mohamed, M., Laurent, M., and Ludwig, H. (2017). Security SLA Based Monitoring in Clouds. In *2017 IEEE International Conference on Edge Computing (EDGE)*, pages 90–97.

[218] Kafka, A. (2017). Apache Kafka. https://kafka.apache.org/documentation/#gettingStarted. (Accessed on 07/03/2019).

[219] Kai, L., Weiqin, T., Liping, Z., and Chao, H. (2013). SCM: A Design and Implementation of Monitoring System for CloudStack. In *Proceedings of the 2013 International Conference on Cloud and Service Computing*, CSC '13, pages 146–151, Washington, DC, USA. IEEE Computer Society.

[220] Kapassa, E., Touloupou, M., Mavrogiorgou, A., and Kyriazis, D. (2018). 5G & SLAs: Automated Proposition and Management of Agreements Towards QoS Enforcement. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–5. IEEE.

[221] Karpagavalli, K. and Saravannan, K. (2017). Strategy Tree and Fuzzy Based Cloud SLA Change Management. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–6. IEEE.

[222] Kassa, D. F. and Nahrstedt, K. (2013). SCDA: SLA-Aware Cloud Datacenter Architecture for Efficient Content Storage and Retrieval. In *Proceedings of the 22Nd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '13, pages 121–122, New York, NY, USA. ACM.

[223] Katsalis, K., Papaioannou, T. G., Nikaein, N., and Tassiulas, L. (2016). SLA-Driven VM Scheduling in Mobile Edge Computing. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 750–757.

[224] Kaur, K., Dhand, T., Kumar, N., and Zeadally, S. (2017). Container-as-a-Service at the Edge: Trade-Off between Energy Efficiency and Service Availability at Fog Nano Data Centers. *IEEE Wireless Communications*, 24(3):48–56.

[225] Kaur, K., Garg, S., Aujla, G. S., Kumar, N., Rodrigues, J. J., and Guizani, M. (2018). Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay. *IEEE Communications Magazine*, 56:44–51.

[226] Kaur, K. and Rai, A. K. (2014). A comparative analysis: Grid, cluster and cloud computing. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(3):5730–5734.

[227] Kearney, K. T., Torelli, F., and Kotsokalis, C. (2010). SLA* An Abstract Syntax for Service Level Agreements. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 217–224.

[228] Keller, A. and Ludwig, H. (2003). The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1):57–81.

[229] Kertesz, A., Kecskemeti, G., and Brandic, I. (2009). An SLA-Based Resource Virtualization Approach for On-demand Service Provision. In *Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing*, VTDC '09, page 27–34, New York, NY, USA. Association for Computing Machinery.

[230] Kertesz, A., Kecskemeti, G., and Brandic, I. (2011). Autonomic SLA-Aware Service Virtualization for Distributed Systems. In *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 503–510.

[231] Kertesz, A., Kecskemeti, G., and Brandic, I. (2014). An Interoperable and Self-Adaptive Approach for SLA-Based Service Virtualization in Heterogeneous Cloud Environments. *Future Generation Computer Systems*, 32:54 – 68. Special Section: The Management of Cloud Systems, Special Section: Cyber-Physical Society and Special Section: Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures.

[232] Kesavaraja, D. and Shenbagavalli, A. (2018). QoE Enhancement in Cloud Virtual Machine Allocation Using Eagle Strategy of Hybrid Krill Herd Optimization. *Journal of Parallel and Distributed Computing*, 118:267–279.

[233] Khalil, A., Mbarek, N., and Togni, O. (2017). Service Level Guarantee Framework for IoT Environments: Full Paper. In *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, IML '17, pages 50:1–50:8, New York, NY, USA. ACM.

[234] Khan, H. M., Chan, G.-Y., and Chua, F.-F. (2016). An Adaptive Monitoring Framework for Ensuring Accountability and Quality of Services in Cloud Computing. In *Proceedings of the 2016 International Conference on Information Networking (ICOIN)*, ICOIN '16, pages 249–253, Washington, DC, USA. IEEE Computer Society.

[235] Khan, R., Khan, S. U., Zaheer, R., and Khan, S. (2012). Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In *2012 10th International Conference on Frontiers of Information Technology*, pages 257–260.

[236] Khare, S., Sun, H., Zhang, K., Gascon-Samson, J., Gokhale, A., Koutsoukos, X., and Abdelaziz, H. (2018). Scalable Edge Computing for Low Latency Data Dissemination in Topic-Based Publish/Subscribe. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 214–227.

[237] Kim, E. C., Song, J. G., and Hong, C. S. (2000). An Integrated CNM Architecture for Multi-Layer Networks with Simple SLA Monitoring and Reporting Mechanism. In *Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP*, pages 993–994.

[238] Klingert, S., Schulze, T., and Bunse, C. (2011). GreenSLAs for the Energy-Efficient Management of Data Centres. In *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking*, e-Energy '11, pages 21–30, New York, NY, USA. ACM.

[239] Kochovski, P., Sakellariou, R., Bajec, M., Drobintsev, P., and Stankovski, V. (2019). An Architecture and Stochastic Method for Database Container Placement in the Edge-Fog-Cloud Continuum. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 396–405.

[240] KOGURE, M. (1983). Quality Function Deployment and CWQC in Japan. *Quality Progress*, pages 26–32.

[241] Kohne, A., Krüger, M., Pfahlberg, M., Spinczyk, O., and Nagel, L. (2017). Financial Evaluation of SLA-Based VM Scheduling Strategies for Cloud Federations. In *Proceedings of the 4th Workshop on CrossCloud Infrastructures & Platforms*, Crosscloud'17, New York, NY, USA. Association for Computing Machinery.

[242] Kohne, A., Pasternak, D., Nagel, L., and Spinczyk, O. (2016). Evaluation of SLA-Based Decision Strategies for VM Scheduling in Cloud Data Centers. In *Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms*, CrossCloud '16, New York, NY, USA. Association for Computing Machinery.

[243] Kolomvatsos, K. and Anagnostopoulos, C. (2019). Multi-Criteria Optimal Task Allocation at the Edge. *Future Generation Computer Systems*, 93:358 – 372.

[244] Kolozali, S., Bermudez-Edo, M., Puschmann, D., Ganz, F., and Barnaghi, P. (2014). A Knowledge-Based Approach for Real-Time IoT Data Stream Annotation and Processing. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, pages 215–222.

[245] Kouki, Y., d. Oliveira, F. A., Dupont, S., and Ledoux, T. (2014). A Language Support for Cloud Elasticity Management. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 206–215.

[246] Kouki, Y., De Oliveira, F. A., Dupont, S., and Ledoux, T. (2014). A Language Support for Cloud Elasticity Management. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 206–215. IEEE.

[247] Kousiouris, G., Aisopos, F., Psychas, A., Varvarigou, T., Domaschka, J., Baur, D., Griesinger, F., Nikolov, V., Lyberopoulos, G., Theodoropoulou, E., et al.

(2017). A Toolkit Based Architecture for Optimizing Cloud Management, Performance Evaluation and Provider Selection Processes. In *2017 International Conference on High Performance Computing & Simulation (HPCS)*, pages 224–232. IEEE.

[248] Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benrernou, S., Brandic, I., Kertész, A., Parkin, M., and Carro, Manue, P. (2013). A Survey on Service Quality Description. *ACM Computing Surveys (CSUR)*, 46(1):1.

[249] Kuebert, R., Gallizo, G., Oberle, K., and Oliveros, E. (2010). Enhancing the SLA Framework of a Virtualized Service Platform by Dynamic Re-negotiation. In *eChallenges e-2010 Conference*, pages 1–8. IEEE.

[250] Kueh, P. J. and Mashaly, M. E. (2017). Load Balancing in Distributed Cloud Data Center Configurations: Performance and Energy-Efficiency. In *Proceedings of the Eighth International Conference on Future Energy Systems*, e-Energy '17, pages 296–301, New York, NY, USA. ACM.

[251] Kumar, P., Singh, P., Chopra, S., Sarna, J. S., and Rawat, K. (2017). Inspection of Cloud Computing Monitoring Tools. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS)*, pages 361–365. IEEE.

[252] Kyriazis, D. (2013). Cloud Computing Service Level Agreements-Exploitation of Research Results. *European Commission Directorate General Communications Networks Content and Technology Unit, Tech. Rep*, 5:29.

[253] Labidi, T., Mtibaa, A., Gaaloul, W., and Gargouri, F. (2017). Ontology-Based SLA Negotiation and Re-negotiation for Cloud Computing. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 36–41. IEEE.

[254] Labidi, T., Mtibaa, A., Gaaloul, W., Tata, S., and Gargouri, F. (2017). Cloud SLA Modeling and Monitoring. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 338–345.

[255] Labidi, T., Mtibaa, A., and Gargouri, F. (2018). Cloud SLA Terms Analysis Based On Ontology. *Procedia Computer Science*, 126:292–301.

[256] Lamanna, D. D., Skene, J., and Emmerich, W. (2003). Slang: A Language for Defining Service Level Agreements. In *NINTH IEEE WORKSHOP ON FUTURE TRENDS OF DISTRIBUTED COMPUTING SYSTEMS, PROCEEDINGS*, pages 100–106. IEEE COMPUTER SOC.

[257] Landi, G., Neves, P. M., Edmonds, A., Metsch, T., Mueller, J., and Crosta, P. S. (2014). SLA Management and Service Composition of Virtualized Applications in Mobile Networking Environments. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8. IEEE.

[258] Larsson, L., Henriksson, D., and Elmroth, E. (2011). Scheduling and Monitoring of Internally Structured Services in Cloud Federations. In *Proceedings of the 2011 IEEE Symposium on Computers and Communications*, ISCC '11, pages 173–178, Washington, DC, USA. IEEE Computer Society.

[259] Lauslahti, K., Mattila, J., and Seppälä, T. (2017). Smart Contracts – How Will Blockchain Technology Affect Contractual Practices? ETLA Reports 68, The Research Institute of the Finnish Economy.

[260] Lee, B.-H., Song, T. G., and Kim, D.-H. (2016). Block Storage Scheduling Based on SLA in Cloud Storage Systems. In *Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory*, EDB '16, page 72–76, New York, NY, USA. Association for Computing Machinery.

[261] Leitner, P., Hummer, W., Satzger, B., Inzinger, C., and Dustdar, S. (2012). Cost-Efficient and Application SLA-Aware Client Side Request Scheduling in an Infrastructure-as-a-Service Cloud. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 213–220.

[262] Lemos, A. L., Daniel, F., and Benatallah, B. (2015). Web Service Composition: A Survey of Techniques and Tools. *ACM Comput. Surv.*, 48(3):33:1–33:41.

[263] Lera, I., Guerrero, C., and Juiz, C. (2019). Availability-Aware Service Placement Policy in Fog Computing Based on Graph Partitions. *IEEE Internet of Things Journal*, 6(2):3641–3651.

[264] Li, B. and Yu, J. (2011). Research and Application on the Smart Home Based on Component Technologies and Internet of Things. *Procedia Engineering*, 15(Supplement C):2087 – 2092. CEIS 2011.

[265] Li, F. (2019). Service Negotiation in a Dynamic IoT Environment. In Liu, X., Mrissa, M., Zhang, L., Benslimane, D., Ghose, A., Wang, Z., Bucchiarone, A., Zhang, W., Zou, Y., and Yu, Q., editors, *Service-Oriented Computing – ICSOC 2018 Workshops*, pages 379–386, Cham. Springer International Publishing.

[266] Li, F. and Clarke, S. (2019). A Context-Based Strategy for SLA Negotiation in the IoT Environment. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 208–213. IEEE.

[267] Li, F., Palade, A., and Clarke, S. (2019). A Model for Distributed Service Level Agreement Negotiation in Internet of Things. In Yangui, S., Bouassida Rodriguez, I., Drira, K., and Tari, Z., editors, *Service-Oriented Computing*, pages 71–85, Cham. Springer International Publishing.

[268] Li, G., Pourraz, F., and Moreaux, P. (2014). PSLA: A PaaS Level SLA Description Language. In *Proceedings of the 2014 IEEE International Conference on Cloud Engineering*, IC2E '14, pages 452–457, Washington, DC, USA. IEEE Computer Society.

[269] Li, H., Gao, X., and Di, Y. (2015a). SLA-Aware Resource Reservation Management in Cloud Workflows. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 4226–4231.

[270] Li, J., Wang, Y., Lin, X., Nazarian, S., and Pedram, M. (2016). Negotiation-Based Resource Provisioning and Task Scheduling Algorithm for Cloud Systems. In *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pages 338–343. IEEE.

[271] Li, L., Dong, J., Zuo, D., and Liu, J. (2018). SLA-Aware and Energy-Efficient VM Consolidation in Cloud Data Centers Using Host States Naive Bayesian Prediction Model. In *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 80–87.

[272] Li, L., Dong, J., Zuo, D., and Wu, J. (2019). SLA-Aware and Energy-Efficient VM Consolidation in Cloud Data Centers Using Robust Linear Regression Prediction Model. *IEEE Access*, 7:9490–9500.

[273] Li, P., Ju, L., Jia, Z., and Sun, Z. (2015b). SLA-Aware Energy-Efficient Scheduling Scheme for Hadoop YARN. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pages 623–628.

[274] Li, Y., Alqahtani, A., Solaiman, E., Perera, C., Jayaraman, P. P., Buyya, R., Morgan, G., and Ranjan, R. (2019). IoT-CANE: A Unified Knowledge Management System for Data-Centric Internet of Things Application Systems. *Journal of Parallel and Distributed Computing*, 131:161 – 172.

[275] Liccardo, L., Rak, M., Di Modica, G., and Tomarchio, O. (2012). Ontology-Based Negotiation of Security Requirements in Cloud. In *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, pages 192–197. IEEE.

[276] Lim, N., Majumdar, S., and Ashwood-Smith, P. (2014). A Constraint Programming-Based Resource Management Technique for Processing MapReduce Jobs with SLAs on Clouds. In *2014 43rd International Conference on Parallel Processing*, pages 411–421.

[277] Lin, Y. and Shen, H. (2017). CloudFog: Leveraging Fog to Extend Cloud Gaming for Thin-Client MMOG with High Quality of Service. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):431–445.

[278] Liu, D., Kanabar, U., and Lung, C.-H. (2013). A Light Weight SLA Management Infrastructure for Cloud Computing. In *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4. IEEE.

[279] Liu, L., Chang, Z., Guo, X., Mao, S., and Ristaniemi, T. (2018). Multiobjective Optimization for Computation Offloading in Fog Computing. *IEEE Internet of Things Journal*, 5(1):283–294.

[280] Liu, X., Wang, Q., Sha, L., and He, Wenbo, P. (2003). Optimal QoS Sampling Frequency Assignment for Real-Time Wireless Sensor Networks. *in In RTSS*.

[281] Liu, X. and Xu, F. (2013). Cloud Service Monitoring System Based on SLA. In *Proceedings of the 2013 12th International Symposium on Distributed Computing and Applications to Business, Engineering & Science*, DCABES '13, pages 137–141, Washington, DC, USA. IEEE Computer Society.

[282] Longo, A., Zappatore, M., and Bochicchio, M. A. (2015a). Service and Contract Composition: A Model and a Tool. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1434–1440. IEEE.

[283] Longo, A., Zappatore, M., and Bochicchio, M. A. (2015b). Service Level Aware - Contract Management. In *Proceedings of the 2015 IEEE International Conference on Services Computing*, SCC '15, pages 499–506, Washington, DC, USA. IEEE Computer Society.

[284] López, P. G., Montresor, A., Epema, D. H. J., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M. P., Felber, P., and Rivière, E. (2015). Edge-Centric Computing: Vision and Challenges. *Computer Communication Review*, 45:37–42.

[285] Lu, C.-T., Chang, C.-W., and Li, J.-S. (2015). VM Scaling Based on Hurst Exponent and Markov Transition with Empirical Cloud Data. *J. Syst. Softw.*, 99(C):199–207.

[286] Lu, K., Roblitz, T., Yahyapour, R., Yaqub, E., and Kotsokalis, C. (2011). QoS-Aware SLA-Based Advanced Reservation of Infrastructure as a Service. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, pages 288–295.

[287] Lu, K., Yahyapour, R., Wieder, P., Yaqub, E., Abdullah, M., Schloer, B., and Kotsokalis, C. (2016). Fault-Tolerant Service Level Agreement Lifecycle Management in Clouds Using Actor System. *Future Gener. Comput. Syst.*, 54(C):247–259.

[288] Lu Tan and Neng Wang (2010). Future Internet: The Internet of Things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages V5–376–V5–380.

[289] Ludwig, H. (2003). Web Services QoS: External SLAs and Internal Policies or: How Do We Deliver What We Promise? In *Fourth International Conference on Web Information Systems Engineering Workshops, 2003. Proceedings.*, pages 115–120.

[290] Ludwig, H., Keller, A., Dan, A., and King, R. (2002). A Service Level Agreement Language for Dynamic Electronic Services. In *Proceedings Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002)*, pages 25–32.

[291] Maarouf, A., El Hamlaoui, M., Marzouk, A., and Haqiq, A. (2015a). Combining Multi-Agent Systems and MDE Approach for Monitoring SLA Violations in the Cloud Computing. In *2015 International Conference on Cloud Technologies and Applications (CloudTech)*, pages 1–6. IEEE.

[292] Maarouf, A., Marzouk, A., and Haqiq, A. (2015b). A Review of SLA Specification Languages in the Cloud Computing. In *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6. IEEE.

[293] Maarouf, A., Marzouk, A., and Haqiq, A. (2015). A review of sla specification languages in the cloud computing. In *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6.

[294] Maarouf, A., Marzouk, A., Haqiq, A., and El Hamlaoui, M. (2014). Towards a MDE Approach for the Establishment of a Contract Service Level Monitoring by Third Party in the Cloud Computing. In *Proceedings of the 2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, SITIS '14, pages 715–720, Washington, DC, USA. IEEE Computer Society.

[295] Mach, W. (2017). A Simulation Environment for WS-Agreement Negotiation Compliant Strategies. In *Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '17, pages 462–471, New York, NY, USA. ACM.

[296] Mach, W. and Schikuta, E. (2012). A Generic Negotiation and Renegotiation Framework for Consumer-Provider Contracting of Web Services. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*, pages 348–351. ACM.

[297] Macias, M. and Guitart, J. (2010). Using Resource-Level Information into Nonadditive Negotiation Models for Cloud Market Environments. In *2010 IEEE Network Operations and Management Symposium-NOMS 2010*, pages 325–332. IEEE.

[298] Macías, M. and Guitart, J. (2011). A Genetic Model for Pricing in Cloud Computing Markets. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 113–118, New York, NY, USA. ACM.

[299] Macias, M. and Guitart, J. (2012). Client Classification Policies for SLA Enforcement in Shared Cloud Datacenters. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, CCGRID '12, pages 156–163, Washington, DC, USA. IEEE Computer Society.

[300] Macías, M. and Guitart, J. (2014). SLA Negotiation and Enforcement Policies for Revenue Maximization and Client Classification in Cloud Providers. *Future Gener. Comput. Syst.*, 41(C):19–31.

[301] Macías, M. and Guitart, J. (2016). Analysis of a Trust Model for SLA Negotiation and Enforcement in Cloud Markets. *Future Gener. Comput. Syst.*, 55(C):460–472.

[302] Madheswari, A. N. et al. (2013). Performance Optimized Routing for SLA Enforcement in Cloud Computing. In *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, pages 689–693. IEEE.

[303] Mahmoud, M. M., Rodrigues, J. J., Saleem, K., Al-Muhtadi, J., Kumar, N., and Korotaev, V. (2018). Towards Energy-aware Fog-enabled Cloud of Things for Healthcare. *Computers & Electrical Engineering*, 67:58 – 69.

[304] Mahmud, M. R., Afrin, M., Razzaque, M. A., Hassan, M. M., Alelaiwi, A., and Alrubaian, M. (2016a). Maximizing Quality of Experience through Context-aware Mobile Application Scheduling in Cloudlet Infrastructure. *Software: Practice and Experience*, 46(11):1525–1545.

[305] Mahmud, R. and Buyya, R. (2016). Fog Computing: A Taxonomy, Survey and Future Directions. *CoRR*, abs/1611.05539.

[306] Mahmud, R. and Buyya, R. (2019). Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. *Fog and edge computing: Principles and paradigms*, pages 1–35.

[307] Mahmud, R., Kotagiri, R., and Buyya, Rajkumar, P. (2016b). Fog Computing: A Taxonomy, Survey and Future Directions. *arXiv preprint arXiv:1611.05539*.

[308] Mahmud, R., Ramamohanarao, K., and Buyya, R. (2018). Latency-Aware Application Module Management for Fog Computing Environments. *ACM Trans. Internet Technol.*, 19(1).

[309] Mahmud, R., Srirama, S. N., Ramamohanarao, K., and Buyya, R. (2019). Quality of Experience (QoE)-aware Placement of Applications in Fog computing Environments. *Journal of Parallel and Distributed Computing*, 132:190 – 203.

[310] Mahmud, R., Srirama, S. N., Ramamohanarao, K., and Buyya, R. (2020). Profit-Aware Application Placement for Integrated Fog Cloud Computing Environments. *Journal of Parallel and Distributed Computing*, 135:177 – 190.

[311] Maiti, P., Shukla, J., Sahoo, B., and Turuk, A. K. (2018). QoS-Aware Fog Nodes Placement. In *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, pages 1–6.

[312] Maity, S. and Chaudhuri, A. (2014). Optimal Negotiation of SLA in Federated Cloud Using Multiobjective Genetic Algorithms. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 269–271. IEEE.

[313] Maiyza, A. I., Hassan, H. A., Sheta, W. M., Sadek, N. M., and Mokhtar, M. A. (2017). End-User's SLA-Aware Consolidation in Cloud Data Centers. In *2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 196–204.

[314] Mancill, T. (2018). Best Practices for Apache Kafka. https://blog.newrelic.com/engineering/kafka-best-practices/. (Accessed on 07/03/2019).

[315] Martinho, R. and Domingos, D. (2014). Quality of Information and Access Cost of IoT Resources in BPMN Processes. *Procedia Technology*, 16:737–744.

[316] Marz, N. and Warren, J. (2015). *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co., Greenwich, CT, USA, 1st edition.

[317] Mashhadi Moghaddam, S., Fotuhi Piraghaj, S., O'Sullivan, M., Walker, C., and Unsworth, C. (2018). Energy-Efficient and SLA-Aware Virtual Machine Selection Algorithm for Dynamic Resource Allocation in Cloud Data Centers. In *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*, pages 103–113.

[318] Maurer, M., Brandic, I., and Sakellariou, R. (2013). Adaptive Resource Configuration for Cloud Infrastructure Management. *Future Gener. Comput. Syst.*, 29(2):472–487.

[319] Mavrogeorgi, N., Gogouvitis, S., Voulodimos, A., Kiriazis, D., Varvarigou, T., Shulman-Peleg, A., and Kolodner, E. K. (2013). Dynamic Rule Based SLA Management in Clouds. In *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, CLOUD '13, pages 964–965, Washington, DC, USA. IEEE Computer Society.

[320] McCall, J. A., Richards, P. K., and Walters, G. F. (1977). Factors in Software Quality. Technical report, GENERAL ELECTRIC CO SUNNYVALE CA.

[321] Mechalikh, C., Taktak, H., and Moussa, F. (2019). A Scalable and Adaptive Tasks Orchestration Platform for IoT. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 1557–1563.

[322] Mehrotra, R., Srivastava, S., Banicescu, I., and Abdelwahed, S. (2016). Towards an Autonomic Performance Management Approach for a Cloud Broker Environment Using a Decomposition-Coordination Based Methodology. *Future Gener. Comput. Syst.*, 54(C):195–205.

[323] Mell, P. and Grance, T. (2011). The NIST Definition of Cloud Computing (Draft). *NIST Special Publication*, 800:145.

[324] Messina, F., Pappalardo, G., Santoro, C., Rosaci, D., and Sarné, G. M. L. (2014). An Agent Based Negotiation Protocol for Cloud Service Level Agreements. In *Proceedings of the 2014 IEEE 23rd International WETICE Conference*, WETICE '14, pages 161–166, Washington, DC, USA. IEEE Computer Society.

[325] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du (2010). Research on the Architecture of Internet of Things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages V5–484–V5–487.

[326] Michael, N., Ramannavar, N., Shen, Y., Patil, S., and Sung, J.-L. (2017). CloudPerf: A Performance Test Framework for Distributed and Dynamic Multi-Tenant Environments. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, ICPE '17, pages 189–200, New York, NY, USA. ACM.

[327] Microsystems, S. (2002). Service Level Agreement in the Data Center. http://www.dei.unipd.it/~rumor/slide_2006/{SLA}.pdf. (Accessed on 06/05/2019).

[328] Mingozzi, E., Tanganelli, G., and Vallati, C. (2014). A Framework for QoS Negotiation in Things-as-a-Service Oriented Architectures. In *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*, pages 1–5. IEEE.

[329] Minh, Q. T., Nguyen, D. T., Van Le, A., Nguyen, H. D., and Truong, A. (2017). Toward Service Placement on Fog Computing Landscape. In *2017 4th NAFOSTED Conference on Information and Computer Science*, pages 291–296.

[330] Mirobi, G. J. and Arockiam, L. (2015). Service Level Management in Cloud Computing. In *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pages 376–387. IEEE.

[331] Mittal, S., Gupta, A., Joshi, K. P., Pearce, C., and Joshi, A. (2017). A Question and Answering System for Management of Cloud Service Level Agreements. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 684–687. IEEE.

[332] Mittal, S., Joshi, K. P., Pearce, C., and Joshi, A. (2015). Parallelizing Natural Language Techniques for Knowledge Extraction from Cloud Service Level Agreements. In *Proceedings of the 2015 IEEE International Conference on Big Data (Big Data)*, BIG DATA '15, pages 2831–2833, Washington, DC, USA. IEEE Computer Society.

[333] Mittal, S., Joshi, K. P., Pearce, C., and Joshi, A. (2016). Automatic Extraction of Metrics from SLAs for Cloud Service Management. In *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pages 139–142. IEEE.

[334] Mohamed, M., Anya, O., Sakairi, T., Tata, S., Mandagere, N., and Ludwig, H. (2016a). The rSLA Framework: Monitoring and Enforcement of Service Level Agreements for Cloud Services. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 625–632. IEEE.

[335] Mohamed, M., Belaïd, D., and Tata, S. (2013). Monitoring and Reconfiguration for OCCI Resources. In *Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science - Volume 01*, CLOUDCOM '13, pages 539–546, Washington, DC, USA. IEEE Computer Society.

[336] Mohamed, M., Belaïd, D., and Tata, S. (2016b). Extending OCCI for Autonomic Management in the Cloud. *J. Syst. Softw.*, 122(C):416–429.

[337] Molina-Jimenez, C., Shrivastava, S., Crowcroft, J., and Gevros, P. (2004). On the Monitoring of Contractual Service Level Agreements. In *Proceedings of the First IEEE International Workshop on Electronic Contracting*, WEC '04, pages 1–8, Washington, DC, USA. IEEE Computer Society.

[338] Moon, H. J., Chi, Y., and Hacigümüs, H. (2010). SLA-Aware Profit Optimization in Cloud Services via Resource Scheduling. In *2010 6th World Congress on Services*, pages 152–153.

[339] Morar, G. A., Ilea, A., Butoi, A., and Silaghi, G. C. (2012). Agent-Based Cloud Resources Negotiation. In *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*, pages 297–300.

[340] Motta, G., You, L., Sfondrini, N., Sacco, D., and Ma, T. (2014). Service Level Management (SLM) in Cloud Computing - Third Party SLM Framework. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, pages 353–358.

[341] Moustafa, S., Elgazzar, K., Martin, P., and Elsayed, M. (2015). SLAM: SLA Monitoring Framework for Federated Cloud Services. In *Proceedings of the 8th International Conference on Utility and Cloud Computing*, UCC '15, pages 506–511, Piscataway, NJ, USA. IEEE Press.

[342] Mubeen, S., Asadollah, S. A., Papadopoulos, A. V., Ashjaei, M., Pei-Breivold, H., and Behnam, M. (2017). Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study. *IEEE Access*, 6:30184–30207.

[343] Mubeen, S., Asadollah, S. A., Papadopoulos, A. V., Ashjaei, M., Pei-Breivold, H., and Behnam, M. (2018). Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study. *IEEE Access*, 6:30184–30207.

[344] Muller, C., Fernandez, A. M. G., Fernandez, P., Martin-Diaz, O., Resinas, M., and Ruiz-Cortes, A. (2018). Automated Validation of Compensable SLAs. *IEEE Transactions on Services Computing*.

[345] Munteanu, V. I., Fortis, T., and Negru, V. (2013). An Evolutionary Approach for SLA-Based Cloud Resource Provisioning. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 506–513.

[346] Mustafa, S., Bilal, K., Malik, S. U. R., and Madani, S. A. (2018). SLA-Aware Energy Efficient Resource Management for Cloud Environments. *IEEE Access*, 6:15004–15020.

[347] Müller, C., Resinas, M., and Ruiz-Cortés, A. (2014). Automated Analysis of Conflicts in WS-Agreement. *IEEE Transactions on Services Computing*, 7(4):530–544.

[348] Naas, M. I., Boukhobza, J., Raipin Parvedy, P., and Lemarchand, L. (2018). An Extension to IFogSim to Enable the Design of Data Placement Strategies. In *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–8.

[349] Nae, V., Prodan, R., and Iosup, A. (2013). SLA-Based Operation of Massively Multiplayer Online Games in Competition-Based Environments. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*, C3S2E '13, pages 104–112, New York, NY, USA. ACM.

[350] Nagin, K., Kassis, A., Lorenz, D., Barabash, K., and Raichstein, E. (2019). Estimating Client QoE from Measured Network QoS. In *Proceedings of the 12th ACM International Conference on Systems and Storage*, SYSTOR '19, pages 188–188, New York, NY, USA. ACM.

[351] Nakamoto, S. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list at https://metzdowd.com*.

[352] Nakamura, Y., Mizumoto, T., Suwa, H., Arakawa, Y., Yamaguchi, H., and Yasumoto, K. (2018). Design and Evaluation of In-Situ Resource Provisioning Method for Regional IoT Services. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2.

[353] Nandi, B. B., Banerjee, A., Ghosh, S. C., and Banerjee, N. (2013). Dynamic SLA Based Elastic Cloud Service Management: A SaaS Perspective. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 60–67.

[354] Nawaz, F., Hussain, O., Hussain, F. K., Janjua, N. K., Saberi, M., and Chang, E. (2019). Proactive Management of SLA Violations by Capturing Relevant External Events in a Cloud of Things Environment. *Future Generation Computer Systems*, 95:26–44.

[355] Nawaz, F., Hussain, O. K., Janjua, N., and Chang, E. (2017). A Proactive Event-Driven Approach for Dynamic QoS Compliance in Cloud of Things. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 971–975, New York, NY, USA. ACM.

[356] Nawaz, F., Janjua, N. K., Hussain, O. K., Hussain, F. K., Chang, E., and Saberi, M. (2018). Event-Driven Approach for Predictive and Proactive Management of SLA Violations in the Cloud of Things. *Future Generation Computer Systems*, 84:78–97.

[357] Nemati, H., Singhvi, A., Kara, N., and Barachi, M. E. (2014). Adaptive SLA-Based Elasticity Management Algorithms for a Virtualized IP Multimedia Subsystem. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 7–11.

[358] Nepal, S. and Zic, J. (2008). A Conflict Neighbouring Negotiation Algorithm for Resource Services in Dynamic Collaborations. In *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*, SCC '08, pages 283–290, Washington, DC, USA. IEEE Computer Society.

[359] Neto, E. C. P., Callou, G., and Aires, F. (2017). An Algorithm to Optimise the Load Distribution of Fog Environments. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1292–1297.

[360] Nguyen, M. and Nguyen, G. (2017). A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness. *Procedia Computer Science (International Conference on Computational Science ICCS 2017)*, 108:365–374.

[361] Nikolow, D., Słota, R., Polak, S., Mitera, D., Pogoda, M., Winiarczyk, P., and Kitowski, J. (2013). Model of QoS Management in a Distributed Data Sharing and Archiving System. *Procedia Computer Science*, 18:100–109.

[362] Nirmala, S. J., Maulik, S., and Bhanu, S. M. S. (2013). SLA Achievement by Negotiation in a Cloud. In *Proceedings of the 6th ACM India Computing Convention*, Compute '13, pages 18:1–18:4, New York, NY, USA. ACM.

[363] Nisar, S. and Bahsoon, R. (2013). An Economics-Driven Approach for Automated SLA Negotiation for Cloud Services Adoption: Aspoc2. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, UCC '13, pages 243–246, Washington, DC, USA. IEEE Computer Society.

[364] Nuseibeh, B. and Easterbrook, S. (2000). Requirements Engineering: A Roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46.

[365] Oberortner, E., Sobernig, S., Zdun, U., and Dustdar, S. (2012). Monitoring Performance-Related QoS Properties in Service-Oriented Systems: A Pattern-Based Architectural Decision Model. In *Proceedings of the 16th European Conference on Pattern Languages of Programs*, page 13. ACM.

[366] Ogino, T., Kitagami, S., Suganuma, T., and Shiratori, N. (2018). A Multi-Agent Based Flexible IoT Edge Computing Architecture Harmonizing Its Control with Cloud Computing. *International Journal of Networking and Computing*, 8(2):218–239.

[367] Oliveira, A. C., Chagas, H., Spohn, M., Gomes, R., and Duarte, B. J. (2014). Efficient Network Service Level Agreement Monitoring for Cloud Computing Systems. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE.

[368] Omezzine, A., Saoud, N. B. B., Tazi, S., and Cooperman, G. (2016). Negotiation Based Scheduling for an Efficient Saas Provisioning in the Cloud. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 33–40. IEEE.

[369] Omezzine, A., Tazi, S., Bellamine, N., Saoud, B., Drira, K., and Cooperman, G. (2015). Towards a Dynamic Multi-Level Negotiation Framework in Cloud Computing. In *2015 International Conference on Cloud Technologies and Applications (CloudTech)*, pages 1–8. IEEE.

[370] Oueis, J., Strinati, Emilio, C., Sardellitti, S., and Barbarossa, Sergio, P. (2015). Small Cell Clustering for Efficient Distributed Fog Computing: A Multi-User Case. *in In: Vehicular Technology Conference (VTC Fall)*.

[371] Paletta, M. and Herrero, P. (2009). A MAS-Based Negotiation Mechanism to Deal with Service Collaboration in Cloud Computing. In *Proceedings of the 2009 International Conference on Intelligent Networking and Collaborative Systems*, INCOS '09, pages 147–153, Washington, DC, USA. IEEE Computer Society.

[372] Pan, J. and McElhannon, J. (2018). Future Edge Cloud and Edge Computing for Internet of Things Applications. *IEEE Internet of Things Journal*, 5(1):439–449.

[373] Pan, L. (2011). Towards a Ramework for Automated Service Negotiation in Cloud Computing. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 364–367. IEEE.

[374] Papadakis-Vlachopapadopoulos, K., González, R. S., Dimolitsas, I., Dechouniotis, D., Ferrer, A. J., and Papavassiliou, S. (2019). Collaborative SLA and Reputation-Based Trust Management in Cloud Federations. *Future Generation Computer Systems*.

[375] Papadopoulos, A., Asadollah, S. A., Ashjaei, M., Mubeen, S., Pei-Breivold, H., and Behnam, M. (2017). SLAs for Industrial IoT: Mind the Gap. In *The 4th International Symposium on Inter-cloud and IoT (ICI 2017)*, pages 75–78. IEEE.

[376] Papazoglou, M. P. and van den Heuvel, W.-J. (2011). Blueprinting the cloud. *IEEE Internet Computing*, 15(6):74–79.

[377] Patel, J., Jindal, V., Yen, I.-L., Bastani, F., Xu, J., and Garraghan, P. (2015). Workload Estimation for Improving Resource Management Decisions in the Cloud. In *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, pages 25–32. IEEE.

[378] Patel, P., Ranabahu, A. H., and Sheth, A. P. (2009). Service Level Agreement in Cloud Computing. *Cloud Workshops at OOPSLA09*.

[379] Patel, S., Park, H., Bonato, P., Chan, L., and Rodgers, M. (2012). A Review of Wearable Sensors and Systems with Application in Rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 9(1):21.

[380] Pavlik, J., Sobeslav, V., and Horalek, J. (2014). Statistics and Analysis of Service Availability in Cloud Computing. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, IDEAS '14, pages 310–313, New York, NY, USA. ACM.

[381] Peng, G., Zhao, J., Li, M., Hou, B., and Zhang, H. (2015). A SLA-Based Scheduling Approach for Multi-Tenant Cloud Simulation. In *2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 600–605.

[382] Perumal, V., Thangavel, J., Ramasamy, S., and Harish, S. (2013). Dynamic Trust Establishment and Amended Window Based Monitoring in Cloud. In *Proceedings of the 2013 International Symposium on Electronic System Design*, ISED '13, pages 162–166, Washington, DC, USA. IEEE Computer Society.

[383] Petcu, D. (2014). A Taxonomy for SLA-Based Monitoring of Cloud Security. In *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, COMPSAC '14, pages 640–641, Washington, DC, USA. IEEE Computer Society.

[384] Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swindon, UK. BCS Learning & Development Ltd.

[385] Pirbhulal, S., Zhang, H., E Alahi, M., Ghayvat, H., Mukhopadhyay, S., Zhang, Y.-T., and Wu, W. (2017). A Novel Secure IoT-Based Smart Home Automation System Using a Wireless Sensor Network. *Sensors*, 17(1):69.

[386] Pires, P. F., Delicato, F. C., Cóbe, R., Batista, T., Davis, J. G., and Song, J. H. (2011). Integrating Ontologies, Model Driven, and CNL in a Multi-Viewed Approach for Requirements Engineering. *Requirements Engineering*, 16(2):133–160.

[387] Pittl, B., Mach, W., and Schikuta, E. (2016a). A Classification of Autonomous Bilateral Cloud SLA Negotiation Strategies. In *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, iiWAS '16, pages 379–388, New York, NY, USA. ACM.

[388] Pittl, B., Mach, W., and Schikuta, E. (2016b). An Implementation of the WS-Agreement Negotiation Standard in CloudSim. In *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 1–4. IEEE.

[389] Pittl, B., Mach, W., and Schikuta, E. (2016c). Bazaar-Extension: A Cloudsim Extension for Simulating Negotiation Based Resource Allocations. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 427–434. IEEE.

[390] Qazi, F., Jhumka, A., and Ezhilchelvan, P. (2017). Towards Automated Enforcement of Cloud SLA. In *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*, UCC '17 Companion, pages 151–156, New York, NY, USA. ACM.

[391] Quevedo, J., Corujo, D., and Aguiar, R. (2014). Consumer Driven Information Freshness Approach for Content Centric Networking. In *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 482–487. IEEE.

[392] Radha, K., Rao, B., Babu, S., Rao, K., Reddy, V., and Saikiran, P. (2015). Service Level Agreements in Cloud Computing and Big Data. *International Journal of Electrical and Computer Engineering*, 5(1):158.

[393] Radziwill, N. (2018). Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World. *The Quality Management Journal*, 25(1):64–65.

[394] Rafique, A., Van Landuyt, D., Reniers, V., and Joosen, W. (2017). Towards an Adaptive Middleware for Efficient Multi-Cloud Data Storage. In *Proceedings of the 4th Workshop on CrossCloud Infrastructures & Platforms*, Crosscloud'17, pages 4:1–4:6, New York, NY, USA. ACM.

[395] Rahim, M. A., Haq, I. U., Durad, H., and Schikuta, E. (2015). Generalized SLA Enforcement Framework Using Feedback Control System. In *2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*, pages 1–6. IEEE.

[396] Rahmani, A., Thanigaivelan, N., Gia, T., Granados, J., Negash, B., Liljeberg, P., and Tenhunen, H, P. (2015). Smart E-Health Gateway: Bringing Intelligence to Internet-of-Things Based Ubiquitous Healthcare Systems. *in 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*.

[397] Rak, M., Venticinque, S., Máhr, T., Echevarria, G., and Esnal, G. (2011). Cloud Application Monitoring: The mOSAIC Approach. In *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, CLOUDCOM '11, pages 758–763, Washington, DC, USA. IEEE Computer Society.

[398] Ranaldo, N. and Zimeo, E. (2013). Capacity-Aware Utility Function for SLA Negotiation of Cloud Services. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, UCC '13, pages 292–296, Washington, DC, USA. IEEE Computer Society.

[399] Ranjan, R. (2014). Streaming Big Data Processing in Datacenter Clouds. *IEEE Cloud Computing*, 1(1):78–83.

[400] Ranjan, R. and Benatallah, B. (2012). Programming Cloud Resource Orchestration Framework: Operations and Research Challenges. *arXiv preprint arXiv:1204.2204*.

[401] Ranjan, R., Harwood, A., and Buyya, R. (2006). SLA-Based Coordinated Superscheduling Scheme for Computational Grids. In *2006 IEEE International Conference on Cluster Computing*, pages 1–8.

[402] Ranjbari, M. and Torkestani, J. A. (2018). A Learning Automata-Based Algorithm for Energy and SLA Efficient Consolidation of Virtual Machines in Cloud Data Centers. *Journal of Parallel and Distributed Computing*, 113:55 – 62.

[403] Ray, B. K., Khatua, S., and Roy, S. (2014). Negotiation Based Service Brokering Using Game Theory. In *2014 Applications and Innovations in Mobile Computing (AIMoC)*, pages 1–8. IEEE.

[404] Razaque, A. and Rizvi, S. S. (2016). Triangular Data Privacy-Preserving Model for Authenticating All Key Stakeholders in a Cloud Environment. *Computers & Security*, 62:328–347.

[405] Rios, E., Mallouli, W., Rak, M., Casola, V., and Ortiz, A. M. (2016). SLA-Driven Monitoring of Multi-Cloud Application Components Using the MUSA Framework. In *2016 IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 55–60. IEEE.

[406] Rizvi, S., Roddy, H., Gualdoni, J., and Myzyri, I. (2017). Three-Step Approach to QoS Maintenance in Cloud Computing Using a Third-Party Auditor. *Procedia Comput. Sci.*, 114(C):83–92.

[407] Rodrigues, C., Lima, S. R., Álvarez Sabucedo, L. M., and Carvalho, P. (2012). An Ontology for Managing Network Services Quality. *Expert Syst. Appl.*, 39(9):7938–7946.

[408] Romano, L., De Mari, D., Jerzak, Z., and Fetzer, C. (2011). A Novel Approach to QoS Monitoring in the Cloud. In *2011 First International Conference on Data Compression, Communications and Processing*, pages 45–51.

[409] Rossi, F., Xavier, M., De Rose, C., Neves Calheiros, R., and Buyya, R. (2016). E-Eco: Performance-Aware Energy-Efficient Cloud Data Center Orchestration. *Journal of Network and Computer Applications*, 78.

[410] Roxburgh, D., Spaven, D., and Gallen, C. (2011). Monitoring as an SLA-Oriented Consumable Service for SaaS Assurance: A Prototype. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 925–939. IEEE.

[411] S, A. and K, C. (2015). Monitoring and Management of Service Level Agreements in Cloud Computing. In *Proceedings of the 2015 International Conference on Cloud and Autonomic Computing*, ICCAC '15, pages 204–207, Washington, DC, USA. IEEE Computer Society.

[412] Sahai, A., Graupner, S., Machiraju, V., and van Moorsel, A. (2003). Specifying and Monitoring Guarantees in Commercial Grids through SLA. In *CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings.*, pages 292–299.

[413] Sakr, S. and Liu, A. (2012). SLA-Based and Consumer-Centric Dynamic Provisioning for Cloud Databases. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 360–367.

[414] Saleh, O., Gropengieβer, F., Betz, H., Mandarawi, W., and Sattler, K.-U. (2013). Monitoring and Autoscaling IaaS Clouds: A Case for Complex Event Processing on Data Streams. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 387–392. IEEE Computer Society.

[415] Sampaio, A. M. and Barbosa, J. G. (2016). Chapter Three - Energy-Efficient and SLA-Based Resource Management in Cloud Data Centers. In Hurson, A. R. and Sarbazi-Azad, H., editors, *Energy Efficiency in Data Centers and Clouds*, volume 100 of *Advances in Computers*, pages 103 – 159. Elsevier.

[416] Sampaio, A. M., Barbosa, J. G., and Prodan, R. (2015). PIASA: A Power and Interference Aware Resource Management Strategy for Heterogeneous Workloads in Cloud Data Centers. *Simulation Modelling Practice and Theory*, 57:142–160.

[417] Sauvanaud, C., Kaâniche, M., Kanoun, K., Lazri, K., and Silvestre, G. D. S. (2018). Anomaly Detection and Diagnosis for Cloud Services: Practical Experiments and Lessons Learned. *Journal of Systems and Software*, 139:84–106.

[418] Scheid, E. J., Rodrigues, B. B., Granville, L. Z., and Stiller, B. (2019). Enabling Dynamic SLA Compensation Using Blockchain-Based Smart Contracts. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 53–61. IEEE.

[419] Schoonjans, A., Van Landuyt, D., Lagaisse, B., and Joosen, W. (2015). On the Suitability of Black-Box Performance Monitoring for SLA-Driven Cloud Provisioning Scenarios. In *Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware*, ARM 2015, pages 6:1–6:6, New York, NY, USA. ACM.

[420] Schulz, F., Michalk, W., Hedwig, M., McCallister, M., Momm, C., Caton, S., Haas, C., Rolli, D., and Tavas, M. (2012). Service Level Management for Service Value Networks. In *Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, COMPSACW '12, pages 51–56, Washington, DC, USA. IEEE Computer Society.

[421] Serrano, D., Bouchenak, S., Kouki, Y., de Oliveira Jr., F. A., Ledoux, T., Lejeune, J., Sopena, J., Arantes, L., and Sens, P. (2016). SLA Guarantees for Cloud Services. *Future Gener. Comput. Syst.*, 54(C):233–246.

[422] Serrano, D., Bouchenak, S., Kouki, Y., Ledoux, T., Lejeune, J., Sopena, J., Arantes, L., and Sens, P. (2013a). Towards QoS-Oriented SLA Guarantees for Online Cloud Services. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 50–57. IEEE.

[423] Serrano, M., Le-Phuoc, D., Zaremba, M., Galis, A., Bhiri, S., and Hauswirth, M. (2013b). Resource Optimisation in IoT Cloud Systems by Using Matchmaking and Self-Management Principles. In Galis, A. and Gavras, A., editors, *The Future Internet*, pages 127–140, Berlin, Heidelberg. Springer Berlin Heidelberg.

[424] Shah, S. Y., Yuan, Z., Lu, S., and Zerfos, P. (2017). Dependency Analysis of Cloud Applications for Performance Monitoring Using Recurrent Neural Networks. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1534–1543. IEEE.

[425] Shahrivari, S. (2014). Beyond Batch Processing: Towards Real-Time and Streaming Big Data. *Computers*, 3(4):117–129.

[426] Shih, Y., Wang, C., and Pang, A. (2019). Fog Computing Service Provision Using Bargaining Solutions. *IEEE Transactions on Services Computing*, pages 1–1.

[427] Shojaiemehr, B., Rahmani, A. M., and Qader, N. N. (2019). A Three-Phase Process for SLA Negotiation of Composite Cloud Services. *Computer Standards & Interfaces*, 64:85–95.

[428] Singh, A. and Viniotis, Y. (2016). An SLA-Based Resource Allocation for IoT Applications in Cloud Environments. In *2016 Cloudification of the Internet of Things (CIoT)*, pages 1–6.

[429] Singh, S., Chana, I., and Buyya, R. (2017). STAR: SLA-Aware Autonomic Management of Cloud Resources. *IEEE Transactions on Cloud Computing*, pages 1–1.

[430] Sipser, M. (1996). Introduction to the Theory of Computation. *ACM Sigact News*, 27(1):99–100.

[431] Skarlat, O., Nardelli, M., Schulte, S., and Dustdar, S. (2017). Towards QoS-Aware Fog Service Placement. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 89–96.

[432] Skene, J. (2007). *Language Support for Service-Level Agreements for Application-Service Provision*. PhD thesis, University of London.

[433] Smith, R. G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

[434] Solaiman, E., Sfyrakis, I., and Molina-Jimenez, C. (2016). A State Aware Model and Architecture for the Monitoring and Enforcement of Electronic Contracts. In *2016 IEEE 18th Conference on Business Informatics (CBI)*, volume 1, pages 55–63. IEEE.

[435] Son, S., Choi, H.-H., Oh, B. T., Kim, S. W., and Kim, B. S. (2017). Cloud SLA Relationships in Multi-Cloud Environment: Models and Practices. In *Proceedings of the 8th International Conference on Computer Modeling and Simulation*, ICCMS '17, pages 1–6, New York, NY, USA. ACM.

[436] Son, S. and Jun, S. C. (2013). Negotiation-Based Flexible SLA Establishment with SLA-Driven Resource Allocation in Cloud Computing. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 168–171. IEEE.

[437] Son, S. and Jun, S. C. (2013). Negotiation-Based Flexible SLA Establishment with SLA-Driven Resource Allocation in Cloud Computing. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 168–171.

[438] Son, S., Kang, D.-J., and Kim, J.-M. (2014). Design Considerations to Realize Automated SLA Negotiations in a Multi-Cloud Brokerage System. In *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 466–468. IEEE.

[439] Son, S. and Sim, K. M. (2012). A Price- and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations. *Trans. Sys. Man Cyber. Part B*, 42(3):713–728.

[440] Soper, D. S., Demirkan, H., and Goul, M. (2016). Identifying Relevant Socio-Theoretic Foundations for Supporting Multi-Issue IT Cloudsourcing Negotiations. *IEEE Access*, 4:8670–8685.

[441] Souza, A. A. D. P. and Netto, M. A. S. (2015). Using Application Data for SLA-Aware Auto-Scaling in Cloud Environments. In *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 252–255.

[442] Souza, V. B., Masip-Bruin, X., Marin-Tordera, E., Ramirez, W., and Sanchez, S. (2016). Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

[443] Spillner, J. and Schill, A. (2009). Dynamic SLA Template Adjustments Based on Service Property Monitoring. In *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, CLOUD '09, pages 183–189, Washington, DC, USA. IEEE Computer Society.

[444] Sprenkels, R. and Pras, A. (2001). Service Level Agreements. *Internet NG D*, 2:7.

[445] Stamatakis, D. and Papaemmanouil, O. (2014). SLA-Driven Workload Management for Cloud Databases. *2014 IEEE 30th International Conference on Data Engineering Workshops*, pages 178–181.

[446] Stamou, K. (2014). Systematic SLA Data Management. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 63–68, New York, NY, USA. ACM.

[447] Stamou, K., Kantere, V., and Morin, J.-H. (2013a). SLA Data Management Criteria. In *2013 IEEE International Conference on Big Data*, pages 34–42. IEEE.

[448] Stamou, K., Kantere, V., Morin, J.-H., and Georgiou, M. (2013b). A SLA Graph Model for Data Services. In *Proceedings of the Fifth International Workshop on Cloud Data Management*, CloudDB '13, pages 27–34, New York, NY, USA. ACM.

[449] Stamou, K., Kantere, V., Morin, J.-H., and Georgiou, M. (2014). SLA Information Management through Dependency Digraphs: The Case of Cloud Data Services. In *Proceedings of the 2014 47th Hawaii International Conference on System Sciences*, HICSS '14, pages 5038–5047, Washington, DC, USA. IEEE Computer Society.

[450] Stanik, A., Koerner, M., and Kao, O. (2015). Service-Level Agreement Aggregation for Quality of Service-Aware Federated Cloud Networking. *IET Networks*, 4(5):264–269.

[451] Stanik, A., Koerner, M., and Lymberopoulos, L. (2014). SLA-Driven Federated Cloud Networking: Quality of Service for Cloud-Based Software Defined Networks. *Procedia Computer Science*, 34:655–660.

[452] Stavrinides, G. L. and Karatza, H. D. (2019). A Hybrid Approach to Scheduling Real-Time IoT Workflows in Fog and Cloud Environments. *Multimedia Tools and Applications*, 78(17):24639–24655.

[453] Su, W., Hu, J., Lin, C., and Shen, S. (2015). SLA-Aware Tenant Placement and Dynamic Resource Provision in SaaS. In *2015 IEEE International Conference on Web Services*, pages 615–622.

[454] Sudan, K., Srinivasan, S., Balasubramonian, R., and Iyer, R. (2012). Optimizing Datacenter Power with Memory System Levers for Guaranteed Quality-of-Service. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, pages 117–126, New York, NY, USA. ACM.

[455] Sun, L., Singh, J., and Hussain, O. K. (2012). Service Level Agreement (SLA) Assurance for Cloud Services: A Survey from a Transactional Risk Perspective. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, MoMM '12, pages 263–266, New York, NY, USA. ACM.

[456] Syed, H. J., Gani, A., Ahmad, R. W., Khan, M. K., and Ahmed, A. I. A. (2017). Cloud Monitoring: A Review, Taxonomy, and Open Research Issues. *Journal of Network and Computer Applications*, 98:11–26.

[457] Tan, W., Sun, Y., Li, L., and Tang, A. (2014). Multivariate Quality Control Chart for Monitoring SLA of Workflow Applications. In *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 667–671. IEEE.

[458] Taneja, M. and Davy, A. (2016). Poster Abstract: Resource Aware Placement of Data Stream Analytics Operators on Fog Infrastructure for Internet of Things Applications. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 113–114.

[459] Taneja, M. and Davy, A. (2017). Resource Aware Placement of IoT Application Modules in Fog-Cloud Computing Paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1222–1228.

[460] Tata, S., Mohamed, M., Sakairi, T., Mandagere, N., Anya, O., and Ludwig, H. (2016). rSLA: A Service Level Agreement Language for Cloud Services. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 415–422. IEEE.

[461] Teshome, A., Rilling, L., and Morin, C. (2016). Including Security Monitoring in Cloud Service Level Agreements. In *2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*, pages 209–210. IEEE.

[462] Teshome, A., Rilling, L., and Morin, C. (2018). Verification for Security Monitoring SLAs in IaaS Clouds: The Example of a Network IDS. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE.

[463] Theilmann, W., Lambea, J., Brosch, F., Guinea, S., Chronz, P., Torelli, F., Kennedy, J., Nolan, M., Zacco, G., Spanoudakis, G., and Stopar, M, P. (2011). Service Level Agreements in Cloud Computing and Big Data. *SLA@ SOI Final Report*.

[464] Torkashvan, M. and Haghighi, H. (2012). CSLAM: A Framework for Cloud Service Level Agreement Management Based on WSLA. In *6th International Symposium on Telecommunications (IST)*, pages 577–585. IEEE.

[465] Toueir, A., Broisin, J., and Sibilla, M. (2013). A Goal-Oriented Approach for Adaptive SLA Monitoring: A Cloud Provider Case Study. In *2nd IEEE Latin American Conference on Cloud Computing and Communications*, pages 53–58. IEEE.

[466] Touloupou, M., Kapassa, E., Symvoulidis, C., Stavrianos, P., and Kyriazis, D. (2019). An Integrated SLA Management Framework in a 5G Environment. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 233–235.

[467] Tran, D., Tran, N., Nguyen, G., and Nguyen, B. M. (2017). A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness. *Procedia Computer Science*, 108:365–374.

[468] Tran, M. Q., Tai Nguyen, D., Le, V. A., Nguyen, H., and Pham, T. V. (2019). Task Placement on Fog Computing Made Efficient for IoT Application Provision. *Wireless Communications and Mobile Computing*, 2019:1–17.

[469] Trapero, R., Modic, J., Stopar, M., Taha, A., and Suri, N. (2017). A Novel Approach to Manage Cloud Security SLA Incidents. *Future Generation Computer Systems*, 72:193–205.

[470] Ullah, K. W. and Ahmed, A. S. (2014). Demo Paper: Automatic Provisioning, Deploy and Monitoring of Virtual Machines Based on Security Service Level Agreement in the Cloud. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 536–537. IEEE.

[471] Unger, T., Leymann, F., Mauchart, S., and Scheibler, T. (2008). Aggregation of Service Level Agreements in the Context of Business Processes. In *2008 12th International IEEE Enterprise Distributed Object Computing Conference*, pages 43–52.

[472] Uriarte, R. B. (2015). *Supporting Autonomic Management of Clouds: Service-Level-Agreement, Cloud Monitoring and Similarity Learning*. PhD thesis, IMT Institute for Advanced Studies Lucca, Italy.

[473] Uriarte, R. B., De Nicola, R., and Kritikos, K. (2018). Towards Distributed SLA Management with Smart Contracts and Blockchain. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 266–271. IEEE.

[474] Uriarte, R. B., de Nicola, R., and Kritikos, K. (2018). Towards Distributed SLA Management with Smart Contracts and Blockchain. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 266–271.

[475] Uriarte, R. B., De Nicola, R., Scoca, V., and Tiezzi, F. (2019). Defining and Guaranteeing Dynamic Service Levels in Clouds. *Future Generation Computer Systems*, 99:27–40.

[476] Uriarte, R. B., Tiezzi, F., and Nicola, R. D. (2014). SLAC: A Formal Service-Level-Agreement Language for Cloud Computing. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 419–426.

[477] Vakilinia, S., Truchan, C., Kempf, J., and Elbiaze, H. (2018). Automated Enforcement of SLA for Cloud Services. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 49–56. IEEE.

[478] Van, H. N., Tran, F. D., and Menaud, J. (2009). SLA-Aware Virtual Resource Management for Cloud Infrastructures. In *2009 Ninth IEEE International Conference on Computer and Information Technology*, volume 1, pages 357–362.

[479] van Solingen, D. R. and Berghout, E. W. (1999). *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*. McGraw-Hill.

[480] Vanlightly, J. (2017). RabbitMQ vs Kafka Part 4 - Message Delivery Semantics and Guarantees — Jack Vanlightly. https://jack-vanlightly.com/blog/2017/12/15/rabbitmq-vs-kafka-part-4-message-delivery-semantics-and-guarantees. (Accessed on 07/04/2019).

[481] Vasisht, D., Kapetanovic, Z., Won, J., Jin, X., Chandra, R., Sinha, S., Kapoor, A., Sudarshan, M., and Stratman, S. (2017). Farmbeats: An IoT Platform for data-Driven Agriculture. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 515–529.

[482] Venugopal, S., Chu, X., and Buyya, R. (2008). A Negotiation Mechanism for Advance Resource Reservations Using the Alternate Offers Protocol. In *2008 16th Interntional Workshop on Quality of Service*, pages 40–49.

[483] Verma, A., Mansuri, A. H., and Jain, N. (2016). Big Data Management Processing with Hadoop MapReduce and Spark Technology: A Comparison. In *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pages 1–4. IEEE.

[484] Villalpando, L. E. B., April, A., and Abran, A. (2016). Cloudmeasure: A Platform for Performance Analysis of Cloud Computing Systems. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 975–979. IEEE.

[485] Villegas, N. M., Müller, H. A., and Tamura, G. (2011). Optimizing Run-Time SOA Governance through Context-Driven SLAs and Dynamic Monitoring. In *2011 International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*, pages 1–10. IEEE.

[486] Voith, T., Oberle, K., Stein, M., Oliveros, E., Gallizo, G., and Kubert, R. (2010). A Path Supervision Framework A Key for Service Monitoring in Infrastructure as a Service (IaaS) Platforms. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 127–130.

[487] Wang, C., Wang, G., Wang, H., Chen, A., and Santiago, R. (2006). Quality of Service (QoS) Contract Specification, Establishment, and Monitoring for Service Level Management. In *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*, pages 49–49.

[488] Wang, M., Jayaraman, P. P., Solaiman, E., Chen, L. Y., Li, Z., Jun, S., Georgakopoulos, D., and Ranjan, R. (2018). A Multi-Layered Performance Analysis for Cloud-Based Topic Detection and Tracking in Big Data Applications. *Future Generation Computer Systems*, 87:580–590.

[489] Wang, M., Ranjan, R., Jayaraman, P. P., Strazdins, P., Burnap, P., Rana, O., and Georgakopulos, D. (2015). A Case for Understanding End-to-End Performance of Topic Detection and Tracking Based Big Data Applications in the Cloud. In *International Internet of Things Summit*, pages 315–325. Springer.

[490] Wang, Y., He, Q., Ye, D., and Yang, Y. (2017). Formulating Criticality-Based Cost-Effective Monitoring Strategies for Multi-Tenant Service-Based Systems. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 325–332. IEEE.

[491] Wang, Y., Lin, X., and Pedram, M. (2014). A Game Theoretic Framework of SLA-Based Resource Allocation for Competitive Cloud Service Providers. In *2014 Sixth Annual IEEE Green Technologies Conference*, pages 37–43.

[492] Wang, Z., Tang, X., and Luo, X. (2011). Policy-Based SLA-Aware Cloud Service Provision Framework. In *2011 Seventh International Conference on Semantics, Knowledge and Grids*, pages 114–121.

[493] Whitmore, A., Agarwal, A., and Da Xu, L. (2015). The Internet of Things—A Survey of Topics and Trends. *Information systems frontiers*, 17(2):261–274.

[494] wikiwiki (2016). Open Data Handbook . http://opendatahandbook.org/glossary/en/terms/machine-readable/. Accessed: 2020-07-08.

[495] Wu, L. and Buyya, R. (2012). Service Level Agreement (SLA) in Utility Computing Systems. In *Performance and dependability in service computing: Concepts, techniques and research directions*, pages 1–25. IGI Global.

[496] Wu, L., Garg, S. K., and Buyya, R. (2011). SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 195–204.

[497] Wu, L., Garg, S. K., and Buyya, R. (2015). Service Level Agreement(SLA) Based SaaS Cloud Management System. In *Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, ICPADS '15, pages 440–447, Washington, DC, USA. IEEE Computer Society.

[498] Wu, L., Garg, S. K., Buyya, R., Chen, C., and Versteeg, S. (2013). Automated SLA Negotiation Framework for Cloud Computing. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 235–244. IEEE.

[499] Wu, L., Garg, S. K., Versteeg, S., and Buyya, R. (2014). SLA-Based Resource Provisioning for Hosted Software-as-a-Service Applications in Cloud Computing Environments. *IEEE Transactions on Services Computing*, 7(3):465–485.

[500] Xiong, K. and Chen, X. (2015). Ensuring Cloud Service Guarantees via Service Level Agreement (SLA)-Based Resource Allocation. In *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops*, pages 35–41.

[501] Xu, J., Ota, K., and Dong, M. (2018). Plug-and-Play for Fog: Dynamic Service Placement in Wireless Multimedia Networks. In *2018 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 490–494.

[502] Xu, Y., Yao, J., Jacobsen, H.-A., and Guan, H. (2017). Cost-Efficient Negotiation over Multiple Resources with Reinforcement Learning. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE.

[503] Yahya, F., Walters, R. J., and Wills, G. B. (2017). Using Goal-Question-Metric (GQM) Approach to Assess Security in Cloud Storage. In Chang, V., Ramachandran, M., Walters, R. J., and Wills, G., editors, *Enterprise Security*, pages 223–240, Cham. Springer International Publishing.

[504] Yao, J. and Ansari, N. (2018). Reliability-Aware Fog Resource Provisioning for Deadline-Driven IoT Services. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

[505] Yao, J. and Ansari, N. (2019a). Energy-Aware Task Allocation for Mobile IoT by Online Reinforcement Learning. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6.

[506] Yao, J. and Ansari, N. (2019b). Fog Resource Provisioning in Reliability-Aware IoT Networks. *IEEE Internet of Things Journal*, 6(5):8262–8269.

[507] Yao, J. and Ansari, N. (2019c). QoS-Aware Fog Resource Provisioning and Mobile Device Power Control in IoT Networks. *IEEE Transactions on Network and Service Management*, 16(1):167–175.

[508] Yao, Z., Papapanagiotou, I., and Callaway, R. D. (2014). SLA-Aware Resource Scheduling for Cloud Storage. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 14–19.

[509] Yaqub, E., Yahyapour, R., Wieder, P., Jehangiri, A. I., Lu, K., and Kotsokalis, C. (2014). Metaheuristics-Based Planning and Optimization for SLA-Aware Resource Management in PaaS Clouds. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 288–297.

[510] Yaqub, E., Yahyapour, R., Wieder, P., Kotsokalis, C., Lu, K., and Jehangiri, A. I. (2014). Optimal Negotiation of Service Level Agreements for Cloud-Based Services through Autonomous Agents. In *Proceedings of the 2014 IEEE International Conference on Services Computing*, SCC '14, pages 59–66, Washington, DC, USA. IEEE Computer Society.

[511] Yin, B., Cheng, Y., Cai, L. X., and Cao, X. (2017). Online SLA-Aware Multi-Resource Allocation for Deadline Sensitive Jobs in Edge-Clouds. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6.

[512] Yoori Oh, Jieun Choi, Eunjung Song, Moonji Kim, and Yoonhee Kim (2016). A SLA-Based Spark Cluster Scaling Method in Cloud Environment. In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4.

[513] Yousefpour, A., Patil, A., Ishigaki, G., Kim, I., Wang, X., Cankaya, H. C., Zhang, Q., Xie, W., and Jue, J. P. (2019). FOGPLAN: A Lightweight QoS-Aware Dynamic Fog Service Provisioning Framework. *IEEE Internet of Things Journal*, 6(3):5080–5096.

[514] Yuan, H., Bi, J., Li, B. H., Chai, X., and Tie, M. (2012). SLA-Based Virtualized Resource Allocation for Multi-Tier Web Application in Cloud Simulation Environment. In *2012 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1681–1685.

[515] Yuan Wei, Son, S. H., and Stankovic, J. A. (2006). RTSTREAM: Real-Time Query Processing for Data Streams. In *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06)*, pages 10 pp.–.

[516] Zappatore, M., Longo, A., and Bochicchio, M. A. (2015). SLA Composition in Service Networks: A Tool for Representing Relationships between SLAs and Contracts. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 1219–1224, New York, NY, USA. ACM.

[517] Zaslavsky, A. B., Perera, C., and Georgakopoulos, D. (2013). Sensing as a Service and Big Data. *CoRR*, abs/1301.0159.

[518] Zeng, X., Garg, S., Wen, Z., Strazdins, P., Wang, L., and Ranjan, R. (2016). SLA-Aware Scheduling of Map-Reduce Applications on Public Clouds. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 655–662.

[519] Zeng, X., Ranjan, R., Strazdins, P., Garg, S. K., and Wang, L. (2015). Cross-Layer SLA Management for Cloud-Hosted Big Data Analytics Applications. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 765–768. IEEE.

[520] Zhang, J., He, Z., Huang, H., Wang, X., Gu, C., and Zhang, L. (2014). SLA Aware Cost Efficient Virtual Machines Placement in Cloud Computing. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pages 1–8.

[521] Zhang, S. and Lee, J. (2019). Double-Spending with a Sybil Attack in the Bitcoin Decentralized Network. *IEEE Transactions on Industrial Informatics*, pages 1–1.

[522] Zhang, S., Yen, I.-L., and Bastani, F. B. (2016). Toward Semantic Enhancement of Monitoring Data Repository. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pages 140–147. IEEE.

[523] Zhang, Y., Liu, H., Lu, Y., and Deng, B. (2013). SLA-Driven State Monitoring for Cloud Services. In *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, pages 428–433. IEEE.

[524] Zhang, Z., Liao, L., Liu, H., and Li, G. (2014). Policy-Based Adaptive Service Level Agreement Management for Cloud Services. In *2014 IEEE 5th International Conference on Software Engineering and Service Science*, pages 496–499. IEEE.

[525] Zhao, L., Sakr, S., and Liu, A. (2013a). A Framework for Consumer-Centric SLA Management of Cloud-Hosted Databases. *IEEE Transactions on Services Computing*, 8(4):534–549.

[526] Zhao, L., Sakr, S., and Liu, A. (2013b). Consumer-Centric SLA Manager for Cloud-Hosted Databases. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM '13, pages 2453–2456, New York, NY, USA. ACM.

[527] Zhao, Y., Calheiros, R., Gange, G., Bailey, J., and Sinnott, R. (2018). SLA-Based Profit Optimization Resource Scheduling for Big Data Analytics-as-a-Service Platforms in Cloud Computing Environments. *IEEE Transactions on Cloud Computing*, pages 1–1.

[528] Zhao, Y., Calheiros, R. N., Bailey, J., and Sinnott, R. (2016). SLA-Based Profit Optimization for Resource Management of Big Data Analytics-as-a-Service Platforms in Cloud Computing Environments. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 432–441.

[529] Zhao, Y., Calheiros, R. N., Gange, G., Ramamohanarao, K., and Buyya, R. (2015). SLA-Based Resource Scheduling for Big Data Analytics as a Service in Cloud Computing Environments. In *2015 44th International Conference on Parallel Processing*, pages 510–519.

[530] Zheng, X., Martin, P., Brohman, K., and Da Xu, L. (2014). Cloud Service Negotiation in Internet of Things Environment: A Mixed Approach. *IEEE Transactions on Industrial Informatics*, 10(2):1506–1515.

[531] Zhihong Yang, Yingzhao Yue, Yu Yang, Yufeng Peng, Xiaobo Wang, and Wenji Liu (2011). Study and Application on the Architecture and Key Technologies for IoT. In *2011 International Conference on Multimedia Technology*, pages 747–751.

[532] Zhou, H., de Laat, C., and Zhao, Z. (2018). Trustworthy Cloud Service Level Agreement Enforcement with Blockchain Based Smart Contract. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 255–260. IEEE.

[533] Zhou, H., Ouyang, X., Ren, Z., Su, J., de Laat, C., and Zhao, Z. (2019). A Blockchain Based Witness Model for Trustworthy Cloud Service Level Agreement Enforcement. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1567–1575.

[534] Zhou, N. and Mohindra, A. (2015). Causality-Driven Performance Monitoring and Scaling Automation for Managed Solutions. In *Proceedings of the 2015 IEEE International Conference on Services Computing*, SCC '15, pages 467–474, Washington, DC, USA. IEEE Computer Society.

[535] Zhu, Z., Bi, J., Yuan, H., and Chen, Y. (2011). SLA Based Dynamic Virtualized Resources Provisioning for Shared Cloud Data Centers. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 630–637.

[536] Zhuang, Z., Ramachandra, H., and Sridharan, B. (2015). SLA-Aware Dynamic CPU Scaling in Business Cloud Computing Environments. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 836–843.

[537] Zukowski, M. (2018). Cloud-Based SQL Solutions for Big Data. *Encyclopedia of Big Data Technologies*, pages 1–7.

Appendix A

Questionnaire

Research aim:
Propose a service level agreement specification for IoT applications from the user/service-consumer perspective.
The purpose of the survey:

- Find out: what should be considered and what should not be considered within the SLA.
- Find out: the expressiveness and generality of the proposed SLA grammar

General questions to find out how strongly the interviewee is interested in taking on a role in specifying an SLA for IoT applications.

- What is your primary role in relation to IoT projects?
  ……
  ……
  ……
  ……
  ……
  ……
  ……
  ……

Use-case scenario for clarification purposes:

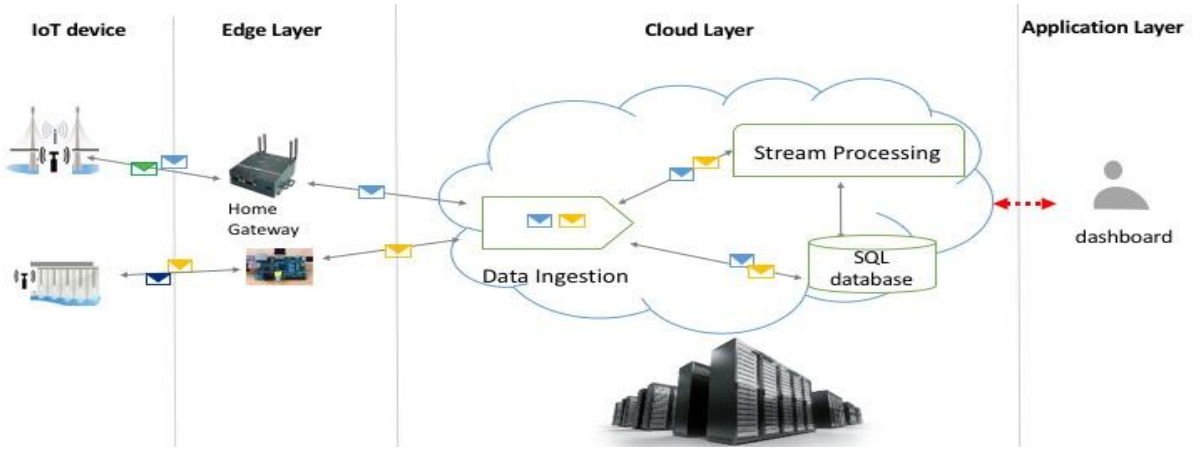

Figure 1 IoT reference architecture

IoT applications are, mostly, time-sensitive applications. Thus, it is important to consider when data needs to be collected, what the next processing step is, and where to process each step. Furthermore, the associated QoS requirements for each step should be specified in an unambiguous way. Therefore, in our work, we focus on end-to-end SLA specification language as a first step to capture user requirements. An end-to-end SLA specification language is a structural syntax that is used to express user requirements for all of the involved software and hardware components within a system/application.

To illustrate how considering an SLA on an end-to-end basis is important, consider a Flood Monitoring System (FMS). In an FMS, it is important to respond immediately and correctly to suspicious events in order to prevent serious damage. FMS requires that real-time data is collected from different types of information, such as from sensors and gauges that measure rainfall levels and the water levels of rivers, respectively. The FMS would then analyse the collected data and indicate any abnormal data patterns (e.g. flood possibility) by comparing the new collected data with the historical/stored data. However, this type of IoT application is time-sensitive, which means that any unpredicted delay in one or more of the workflow activities (e.g. collecting, transferring, ingesting, analysing, etc.) will affect the accuracy and suitability of the actions taken. This example shows how the performance of FMS applications relies not only on the functionality but also on the quality of the offered services across Edge and/or Cloud computing environments. Undoubtedly, SLAs need to be observed across all layers of the Cloud and Edge – for example, at which rate data should be collected, transferred and ingested; how fast and accurate the analysis should be, etc.

Therefore, the data collected from IoT devices should be accurate and up to date. In addition, it is important to minimise the latency of data pre-processing (for filtering purposes), which can be

performed using, but not limited to, a raspberry pi or edge servers. If there is a need to perform some analysis and to compare incoming data with a historical and predefined model, then the application of certain services should be considered, as well as their Service Level Objective (SLO) constraints. A service can apply machine-learning algorithms with a high accuracy requirement, stream processing under low latency constraints and/or batch processing with high throughput. The analysis results can be stored using SQL databases. Unstructured data such as images of collapsed bridges can be stored using NoSQL databases.

In our grammar, we used the "workflow activities" concept to hold the list of involved activity – for example, in the above use case, 'collect rain level data' matches the 'capture event of interest' activity in our grammar. Each activity is associated with a service(s) and an infrastructure resource to deploy the service(s). Both the service and the infrastructure resource have their SLO constraints as well as their own configuration requirements (see Figure 2).

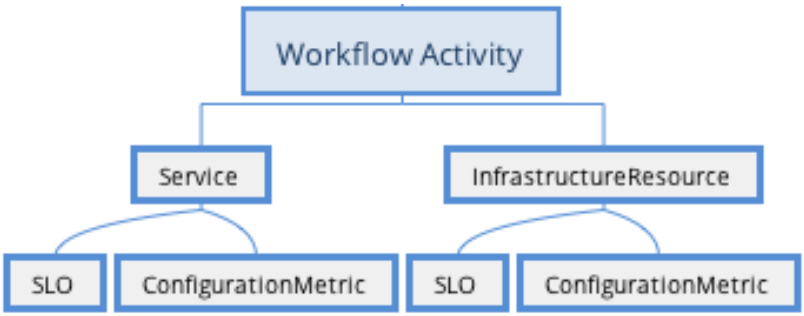

Figure 2 Conceptual mapping to reflect the relationship between main concepts

For further illustration, in the FMS, the 'capture event of interest' activity requires a sensing service. The sensing service has SLO constraints such as constraints on the required level of data freshness and data accuracy. The sensing service will be deployed/hosted on an IoT device. Therefore, it is important to consider the requirements of The IoT device such as its type (e.g., sensor, RFID), the mobility of the device (e.g., fixed or mobile), the communication mechanism and battery life. The same will apply for the "Filter captured event of interest" activity, which will be done at the edge computing layer, using certain devices such as a mobile phone, raspberry pi or server. Each one of these devices has its computational capabilities such as CPU speed and memory size. Furthermore, to perform the "large data analysis on fly" activity, a stream processing service can be applied with certain requirement constraints such as low latency and certain configuration requirements, such as specifying window type: time-based window or event-based window. The stream processing service can be deployed on the Cloud, so certain requirements related to the required resources from the Cloud can be specified, such as number of VMs and the acceptable percentage of CPU utilisation.

Question 1: Considering the following predefined list of <u>workflow activities</u>

- ☐ Capture event of interest (EoI)
- ☐ Examine the Captured EoI on fly
- ☐ Filter Captured EoI
- ☐ Aggregate the Captured EoI
- ☐ Ingest Data from one or more data resources
- ☐ Large-Scale Real-time data analysis
- ☐ Large-Scale Historical data analysis
- ☐ Apply machine learning approach
- ☐ Store Structured Data
- ☐ Store Unstructured Data

1. Does the predefined list of workflow activities cover your IoT project's workflow activities? Answer : Yes or No

   ……

2. Could you please "tick" the activities that you believe will be involved/part of your application/project.

   ……

3. Do you think more workflow activities should be included?

   Answer: Yes or no

   ……

4. if your answer in 3 is "yes", could you please list the workflow activities that you suggest be considered

……

……

……

……

……

……

……

Question 2: Considering the following predefined list of <u>computing layers</u>

☐ IoT device layer
☐ Edge Computing Layer
☐ Cloud Computing Layer

1. Do you agree that it is necessary to allow the IoT administrators to specify their requirements at IoT device, Edge Computing and Cloud Computing layers? Answer: Yes or No

   ……

2. Could you please "tick" the computing layer that you believe will be involved/part of your application/project.

   ……

3. Do you think more computing layers should be included?

   Answer: Yes or no

   ……

4. If your answer in 3 is "yes", could you please list the other computing layers that you suggest be considered

……

……

……

……

……

……

……

Question 3: Considering the following predefined list of <u>services</u>

- ☐ Sensing Service
- ☐ Networking Service
- ☐ Ingestion Service
- ☐ Stream Processing Service
- ☐ Batch Processing Service
- ☐ SQL Database Service
- ☐ NoSQL Database Service
- ☐ Machine Learning Service

1. Does the predefined list of services cover your IoT project's services? Answer : Yes or No

   ……

2. Could you please "tick" the services that you believe will be involved/part of your application/project.

   ……

3. Do you think more services should be included?

   Answer: Yes or no

   ……

4. If your answer in 3 is "yes", could you please list the services that you suggest be considered

……

……

……

……

……

……

……

Question 4: Considering the following predefined list of vocabulary to express consumer requirements *for IoT devices*

☐ Type of device (sensor; RFID tag)
☐ Number of devices
☐ Mobility of device
☐ Communication Mechanism
☐ Communication technology
☐ Battery life
☐ Warranty period
☐ Data integrity

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***IoT devices.***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Question 5: Considering the following predefined list of vocabulary to express consumer requirements of *for Edge Computing*

☐ Type of device (Mobile, Raspberry Pi, Server)
☐ Number of Devices
☐ mobility of device
☐ Communication Mechanism with IoT
☐ Communication Technology with IoT
☐ Communication Mechanism with Cloud
☐ Communication Technology with Cloud
☐ Encryption Support
☐ Compression Support
☐ Storage Size
☐ Memory Size
☐ CPU Speed
☐ Data integrity

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***Edge Computing.***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Question 6: Considering the following predefined list of vocabulary to express consumer requirements *for Cloud Computing*

☐ Availability
☐ CPU utilisation
☐ Response Time
☐ Outage Length
☐ Throughput
☐ Storage Bandwidth
☐ Storage Size
☐ Memory Size
☐ Network Bandwidth
☐ vCPU Capacity
☐ vCPU limit per VM
☐ No Of vCPU
☐ No of core per vCPU
☐ Vertical scale down limit
☐ Vertical scale up limit
☐ Horizontal scale up limit
☐ Horizontal scale down limit
☐ input/output Storage operations
☐ Replication factor
☐ Data integrity

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***Cloud Computing.***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Question 7: Considering the following predefined list of vocabulary to express consumer requirements of <u>sensing services</u>

- ☐ Data Freshness
- ☐ Sample Rate
- ☐ Data accuracy
- ☐ Data integrity

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for **<u>sensing services</u>**.
2. Do you think there are more vocabulary should be included? Answer: Yes or no

   ……
3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

   ……

   ……

Question 8: Considering the following predefined list of vocabulary to express consumer requirements of <u>networking services</u>

- ☐ Gateway Throughput
- ☐ Gateway Latency
- ☐ Network Throughput
- ☐ Network Latency
- ☐ Data integrity
- ☐ Size of data-in/data-in rate
- ☐ Size of data-out/data-out rate
- ☐ Time Interval to send data/Publish rate
- ☐ Storage/buffer size
- ☐ Link Bandwidth

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for **<u>networking services</u>**.

2. Do you think there more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

   ……

   ……

Question 9: Considering the following predefined list of vocabulary to express consumer requirements *for <u>ingestion services</u>*

- ☐ Throughput
- ☐ Latency
- ☐ Data integrity
- ☐ Size of data-in/data-in rate
- ☐ Size of data-out/data-out rate
- ☐ Time Interval to send data/Publish rate
- ☐ Storage size
- ☐ Data retention time limit
- ☐ Replication factor
- ☐ Compression/Decompression support
- ☐ Data Encryption Support
- ☐ Delivery Guarantee Mechanism

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***<u>Ingestion services.</u>***
2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

   ……

Question 10: Considering the following predefined list of vocabulary to express consumer requirements *for <u>machine learning services</u>*

☐ Accuracy
☐ Data integrity
☐ Class of Machine learning
☐ Name of Machine Learning Algorithm
☐ Way to run Machine Learning Algorithm

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***<u>Machine Learning services.</u>***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Question 11: Considering the following predefined list of vocabulary to express consumer requirements *for <u>stream processing services</u>*

- ☐ Throughput
- ☐ Latency
- ☐ Data Completeness
- ☐ Miss Ratio
- ☐ Data integrity
- ☐ Time-based window size
- ☐ Event-based window size
- ☐ Sliding window
- ☐ Tumbling window
- ☐ Micro Batch Size
- ☐ Data Arrival Rate
- ☐ Write Capacity
- ☐ Read Capacity
- ☐ Replication factor
- ☐ Total Number of Queries
- ☐ Number of Queries per Stream
- ☐ Compression/Decompression Support
- ☐ Data Encryption Support

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***<u>Stream Processing services.</u>***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Question 12: Considering the following predefined list of vocabulary to express consumer requirements *for <u>batch processing services</u>*

- ☐ Throughput
- ☐ Latency
- ☐ Data integrity
- ☐ Data Arrival Rate
- ☐ Batch Size Limit
- ☐ Read Capacity
- ☐ Write Capacity
- ☐ Process running frequency
- ☐ Max Memory of Map Task
- ☐ Max Memory of Reduce Task
- ☐ No of Mapper Limit
- ☐ No of Reducer Limit
- ☐ No of Batch Jobs
- ☐ Compression/Decompression Support
- ☐ Data Encryption Support

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***<u>Batch Processing services.</u>***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Question 13: Considering the following predefined list of vocabulary to express consumer requirements *for <u>SQL database services</u>*

- ☐ Throughput
- ☐ Response Time
- ☐ Data integrity
- ☐ Read Error Rate
- ☐ Cache Hit Ratio
- ☐ Write Error Rate
- ☐ Read Capacity
- ☐ Write Capacity
- ☐ Compression/Decompression Support
- ☐ Data Encryption Support
- ☐ Replication factor

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***<u>SQL database services.</u>***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Question 14: Considering the following predefined list of vocabulary to express consumer requirements *for NoSQL database services*

- ☐ Type of NoSQL
- ☐ Throughput
- ☐ Data integrity
- ☐ Response Time
- ☐ Read Error Rate
- ☐ Cache Hit Ratio
- ☐ Write Error Rate
- ☐ Read Capacity
- ☐ Write Capacity
- ☐ Compression/Decompression Support
- ☐ Data Encryption Support
- ☐ Replication factor
- ☐ Replication factor

1. Considering the predefined list of SLO metrics and configuration requirements of sensing services, could you please "tick" the vocabularies that you believe will be used to express your requirements for ***NoSQL database services.***

2. Do you think more vocabulary should be included? Answer: Yes or no

   ……

3. If your answer in 2 is "yes", could you please list the vocabularies that you suggest be considered

   ……

   ……

   ……

   ……

   ……

Appendix B

SLA Specification for RHMS

```
{
[ {
  "appType" : "Remote Health Monitoring Service",
  "startDate" : "Thu Sep 12 12:14:19 BST 2019",
  "endDate" : "Wed Sep 30 12:14:19 BST 2020",
  "sloAtApplicationLevel" : [ {
    "qosMetric" : "Availability",
    "priority" : "High",
    "requiredLevel" : "greater than",
    "value" : "99.0",
    "unit" : "% per month"
  }, {
    "qosMetric" : "Outage Length",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "10",
    "unit" : "seconds per day"
  }, {
    "qosMetric" : "Response Time",
    "priority" : "High",
    "requiredLevel" : "greater than",
    "value" : "10",
    "unit" : "milliseconds"
  } ,{
```

```
"qosMetric" : "Cost/Price",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "100.0",
    "unit" : "$ per month"
    }
],
  "slaid" : " Remote Health Monitoring Service Thu Sep 12 12:14:19
     BST 2019Wed Sep 30 12:14:19 BST 2020j"
} ,{

  "activityName" : "Capture Event of Interest(EoI) ",
  "slo" : [ {
    "qosMetric" : "Data Freshness",
    "priority" : "High ",
    "requiredLevel" : " equals to ",
    "value" : "5 ",
    "unit" : "milliseconds "
  } ],
  "configurationRequirement" : [ {
    "configurationFeature" : "Sample Rate",
    "priority" : "High ",
    "requiredLevel" : " equals to",
    "value" : "5 ",
    "unit" : "kHz "
  }],
  "resourceSpecification" : {
    "resourceLayer" : "IoT Resource",
      "configurationRequirement" : [
      {
      "configurationFeature" : "deviceType",
       "value" : "Sensors "
      },
         {
      "configurationFeature" : "numberOfDevices",
```

```
        "value" : "6 "
       },
        {
       "configurationFeature" : "mobilityOfDevice",
        "value" : "mobile "
       },
        {
       "configurationFeature" : "communicationMechanism",
        "value" : "push "
       },
        {
       "configurationFeature" : "communicationTechnology",
        "value" : "WiFi "
       },
       {
       "configurationFeature" : "cpuCapacity",
        "value" : "1.6 ",
        "unit"  :"GHz"
       },
       {
       "configurationFeature" : "memorySize",
        "value" : "1 ",
        "unit"  :"GB"

       } ]
   }
}, {
"activityName" : "Examine Captured EoI",
  "slo" : [ {
    "qosMetric" : "Gateway Throughput",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "98 ",
    "unit" : "Kbps "
  }, {
```

```
94      "qosMetric" : "Gateway Latency",
95      "priority" : "High ",
96      "requiredLevel" : " less than ",
97      "value" : "1 ",
98      "unit" : "milliseconds "
99    }, {
100     "qosMetric" : "Network Throughput",
101     "priority" : "High ",
102     "requiredLevel" : "greater than ",
103     "value" : "99 ",
104     "unit" : "Mbps "
105   } ],
106   "configurationRequirement" : [ {
107     "configurationFeature" : "Size of data-out",
108     "priority" : "High ",
109     "requiredLevel" : "greater than ",
110     "value" : "8 ",
111     "unit" : "kB "
112   },
113   {
114     "qosMetric" : "Size of data-in",
115     "priority" : "High ",
116     "requiredLevel" : "greater than ",
117     "value" : "10 ",
118     "unit" : "kB "
119   },
120   ,{
121     "configurationFeature" : "Data Encryption Support",
122     "value" : "true "
123   },
124   {
125     "configurationFeature" : "Data Compression Support",
126     "value" : "false "
127   }],
128     "resourceSpecification" : {
```

```
129     "resourceLayer" : "Edge Resource",
130
131          "configurationRequirement" : [
132       {
133       "configurationFeature" : "deviceType",
134        "value" : "Raspberry  Pi "
135       },
136          {
137       "configurationFeature" : "numberOfDevices",
138        "value" : "1 "
139       },
140        {
141       "configurationFeature" : "mobilityOfDevice",
142        "value" : "fixed "
143       },
144        {
145       "configurationFeature" : "communicationMechanismWithIoTDevice
            ",
146        "value" : "push "
147       },
148        {
149       "configurationFeature" : "
            communicationTechnologyWithIoTDevice",
150        "value" : "WiFi "
151       },
152         {
153       "configurationFeature" : "communicationMechanismWithCloud",
154        "value" : "push "
155       },
156        {
157       "configurationFeature" : "communicationTechnologyWithClould",
158        "value" : "WiFi "
159       },
160       {
161       "configurationFeature" : "cpuCapacity",
```

```
        "value" : "1.6 ",
        "unit"  :"GHz"
      },
      {
      "configurationFeature" : "memorySize",
        "value" : "1 ",
        "unit"  :"GB"

      } ]
    ]
  }
},

 {
"activityName" : "Analyse Small-Scale realtime data ",
  "slo" : [ {
    "qosMetric" : "Throughput",
    "priority" : "high ",
    "requiredLevel" : "greater than ",
    "value" : "99 ",
    "unit" : "Mbps "
  } ],
  "configurationRequirement" : [ {
    "configurationFeature" : "Miss Ratio",
    "value" : " less than ",
    "requiredLevel" : "high ",
    "unit" : "1 "
  },{
    "configurationFeature" : "Window Size",
    "requiredLevel" : " equals to ",
    "value" : "5 ",
    "unit" : "milliseconds "
  }, {
    "configurationFeature" : "Sliding Window",
    "requiredLevel" : " equals to ",
```

```
    "value" : "3 ",
    "unit" : "milliseconds "
  }, {
    "configurationFeature" : "Replication factor",
    "requiredLevel" : " equals to ",
    "value" : "2 "
  }, {
    "configurationFeature" : "Total Number of Query",
    "requiredLevel" : " equals to ",
     "value" : "3 "
  }, {
    "configurationFeature" : "Compression/Decompression Support",
    "value" : "false "
  }, {
    "configurationFeature" : "Data Encryption Support",
    "value" : "true "
  } ],
  "resourceSpecification" : {
    "resourceLayer" : "Edge Resource",
       "slo" : [ {
      "qosMetric" : "Availability",
      "priority" : "high ",
      "requiredLevel" : "greater than ",
      "value" : "99 ",
      "unit" : "% "
    },{
    "qosMetric" : "Outage Length",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "3",
    "unit" : "seconds per day"
  } ],
         "configurationRequirement" : [
      {
      "configurationFeature" : "deviceType",
```

```
232         "value" : "server "
233       },
234          {
235       "configurationFeature" : "numberOfDevices",
236         "value" : "1 "
237       },
238         {
239       "configurationFeature" : "mobilityOfDevice",
240         "value" : "fixed "
241       },
242         {
243       "configurationFeature" : "communicationMechanismWithIoTDevice
            ",
244         "value" : "push "
245       },
246         {
247       "configurationFeature" : "
            communicationTechnologyWithIoTDevice",
248         "value" : "WiFi "
249       },
250          {
251       "configurationFeature" : "communicationMechanismWithCloud",
252         "value" : "push "
253       },
254         {
255       "configurationFeature" : "communicationTechnologyWithClould",
256         "value" : "WiFi "
257       },
258       {
259       "configurationFeature" : "cpuCapacity",
260         "value" : "1.6 ",
261         "unit"  :"GHz"
262       },
263       {
264       "configurationFeature" : "memorySize",
```

```
265         "value" : "1 ",
266         "unit"  :"GB"
267
268       } ]
269   }
270 },
271
272  {
273 "activityName" : "Analyse Large-Scale realtime data ",
274   "slo" : [ {
275     "qosMetric" : "Throughput",
276     "priority" : "high ",
277     "requiredLevel" : "greater than ",
278     "value" : "99 ",
279     "unit" : "Mbps "
280   },{
281     "qosMetric" : "Miss Ratio",
282     "priority" : "high ",
283     "value" : " less than ",
284     "requiredLevel" : "high ",
285     "unit" : "1 "
286   }
287    ],
288   "configurationRequirement" : [  {
289     "configurationFeature" : "Window Size",
290     "requiredLevel" : " equals to ",
291     "value" : "30 ",
292     "unit" : "milliseconds "
293   }, {
294     "configurationFeature" : "Sliding Window",
295     "requiredLevel" : " equals to ",
296     "value" : "10 ",
297     "unit" : "milliseconds "
298   }, {
299     "configurationFeature" : "Replication factor",
```

```
    "requiredLevel" : " equals to ",
    "value" : "2 "
  }, {
    "configurationFeature" : "Total Number of Query",
    "requiredLevel" : " equals to ",
     "value" : "3 "
  }, {
    "configurationFeature" : "Compression/Decompression Support",
    "value" : "false "
  }, {
    "configurationFeature" : "Data Encryption Support",
    "value" : "true "
  } ],
  "resourceSpecification" : {
    "resourceLayer" : "Cloudresource",
     "slo" : [ {
      "qosMetric" : "Availability",
      "priority" : "high ",
      "requiredLevel" : "greater than ",
      "value" : "99 ",
      "unit" : "% "
    }, {
      "qosMetric" : "CPU Utilization",
      "priority" : "high ",
      "requiredLevel" : "greater than ",
      "value" : "80 ",
      "unit" : "% "
    } ],
       "configurationRequirement" : [ {
      "configurationFeature" : "Memory Size",
      "value" : "40000 ",
      "requiredLevel" : "greater than ",
      "unit" : "KB "
    }, {
      "configurationFeature" : "vCPU Capacity",
```

```
335       "value" : "44800",
336       "unit" : "GHz Xeon "
337     }, {
338       "configurationFeature" : "Hypervisor",
339       "value" : "Xen "
340     }, {
341       "configurationFeature" : "OS Type",
342       "value" : "Linux Ubuntu "
343     } , {
344       "configurationFeature" : "Tenancy Type",
345       "value" :  "single tenant "
346     }, {
347       "configurationFeature" : "Backup Support",
348       "value" : "true "
349     }, {
350       "configurationFeature" : "Non acceptaple Geographical
            location",
351       "value" : " Hong Kong "
352     }, {
353       "configurationFeature" : "Storage Type",
354       "value" : "HDD(Storage Area Network) "
355     }, {
356       "configurationFeature" : "Hypervisor",
357       "value" : "Xen "
358     }, {
359       "configurationFeature" : "OS Type",
360       "value" : : "Unix "
361     }, {
362       "configurationFeature" : "Geographical location",
363       "value" : "USA "
364     } ]
365   }
366 }, {
367 "activityName" : "Apply machine learing algorithm",
368   "slo" : [ {
```

```
    "qosMetric" : "Accuracy",
    "priority" : "high ",
    "requiredLevel" : "greater than ",
    "value" : "98 ",
    "unit" : "% "
  } ],
  "configurationRequirement" : [ {
    "configurationFeature" : "Name of ML Algorithm ",
    "value" : "neural network "
  }, {
    "configurationFeature" : "Way to Run",
    "value" : " MapReduce "
  } ],
  "resourceSpecification" : {
    "resourceLayer" : "Cloudresource",
    "slo" : [ {
      "qosMetric" : "Availability",
      "priority" : "high ",
      "requiredLevel" : "greater than ",
      "value" : "99 ",
      "unit" : "% "
    },{
    "qosMetric" : "Outage Length",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "3",
    "unit" : "seconds per day"
  }, {
      "qosMetric" : "CPU Utilization",
      "priority" : "high ",
      "requiredLevel" : "greater than ",
      "value" : "80 ",
      "unit" : "% "
    } ],
    "configurationRequirement" : [ {
```

```
      "configurationFeature" : "storage Size",
      "value" : "99 ",
      "requiredLevel" : "greater than ",
      "unit" : "TB "
    },{
      "configurationFeature" : "Memory Size",
      "value" : "40000 ",
      "requiredLevel" : "greater than ",
      "unit" : "KB "
    }, {
      "configurationFeature" : "vCPU Capacity",
      "value" : "44800",
      "unit" : "GHz Xeon "
    }]
  }
}, {
"activityName" : "Store unstructured data ",
  "slo" : [ {
    "qosMetric" : "Read Error Rate",
    "priority" : "% ",
    "requiredLevel" : "1 ",
    "value" : " less than ",
    "unit" : "5 "
  }, {
    "qosMetric" : "Cache Hit Ratio",
    "priority" : "high ",
    "requiredLevel" : " less than ",
    "value" : "5 ",
    "unit" : "% "
  }, {
    "qosMetric" : "Write Error Rate",
    "priority" : "high ",
    "requiredLevel" : " less than ",
    "value" : "5 ",
    "unit" : "% "
```

```
439   },  {
440     "qosMetric" : "Response Time",
441     "priority" : "high ",
442     "requiredLevel" : " less than ",
443     "value" : "1 ",
444     "unit" : "miliseconds "
445   } ],
446   "configurationRequirement" : [ {
447     "configurationFeature" : "Write Capacity",
448     "value" : "99 ",
449     "requiredLevel" : "greater than ",
450     "unit" : "tuples/sec "
451   },
452    {
453     "configurationFeature" : "Read Capacity",
454     "requiredLevel" : "greater than ",
455     "value" : "99 ",
456     "unit" : "tuples/sec "
457   }, {
458     "configurationFeature" : "Type of NOSQL",
459     "value" : " graph "
460   }, {
461       "configurationFeature" : "Back-Up",
462       "value" : "true "
463     } , {
464     "configurationFeature" : "Replication factor",
465     "value" : "3 ",
466     "requiredLevel" : " equals to "
467   }, {
468     "configurationFeature" : "Data Encryption Support",
469     "value" : "true "
470   }, {
471     "configurationFeature" : "Compression/Decompression Support",
472     "value" : "true "
473   }],
```

```
  "resourceSpecification" : {
    "resourceLayer" : "Cloudresource",
    "slo" : [ {
      "qosMetric" : "Availability",
      "priority" : "high ",
      "requiredLevel" : "greater than ",
      "value" : "99 ",
      "unit" : "% "
    } ],
    "configurationRequirement" : [ {
      "configurationFeature" : "storage Size",
      "value" : "99 ",
      "requiredLevel" : "greater than ",
      "unit" : "TB "
    }, {
      "configurationFeature" : "Storge Type",
      "value" : "SAN-HDD(Storage area network) "
    }, {
      "configurationFeature" : "Geographical location",
      "value" : "USA "
    } ]
  }

}
]
}
```

Appendix C

SLA for Case study 1 in Chapter 8

```
[  {
  "appType" : "Remote Health Monitoring Service",
  "startDate" : "Wed Nov 21 00:00:00 GMT 2018",
  "endDate" : "Thu Nov 21 10:35:46 GMT 2019",
  "sloAtApplicationLevel" : [ {
    "qosMetric" : "Response Time",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "1000",
    "unit" : "milliseconds"
  }, {
    "qosMetric" : "Cost/Price",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "3000.0",
    "unit" : "$ per contract period"
  }],
  "slaid" : "Remote Health Monitoring Service Wed Nov 21 00:00:00
     GMT 2018Thu Nov 21 10:35:46 GMT 2019d"
} ,{
    "activityName" : "Capture Event of Interest(EoI) ",
    "slo" : [ {
    "qosMetric" : "Data Freshness",
```

```
    "priority" : "High ",
    "requiredLevel" : "late than real time by ",
    "value" : "2 ",
    "unit" : "milliseconds "
  }, {
    "qosMetric" : "Sample Rate",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "3 ",
    "unit" : "kHz "
  } ],
  "resourceSpecification" : {
   "configurationRequirement" : [
      {
      "configurationFeature" : "deviceType",
       "value" : "Sensors "
      },
         {
      "configurationFeature" : "numberOfDevices",
       "value" : "6 "
      },
       {
      "configurationFeature" : "mobilityOfDevice",
       "value" : "mobile "
      },
       {
      "configurationFeature" : "communicationMechanism",
       "value" : "push "
      },
       {
      "configurationFeature" : "communicationTechnology",
       "value" : "WiFi "
      },
      {
      "configurationFeature" : "cpuCapacity",
```

```
        "value" : "1.6 ",
        "unit"  :"GHz"
      },
      {
      "configurationFeature" : "memorySize",
        "value" : "1 ",
        "unit"  :"GB"

      } ]
  }
}, {
    "activityName" : "Examine Captured EoI",
    "slo" : [ {
    "qosMetric" : "Gateway Throughput",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "6 ",
    "unit" : "Mbps "
  }, {
    "qosMetric" : "Network Latency",
    "priority" : "High ",
    "requiredLevel" : " less than",
    "value" : "2 ",
    "unit" : "milliseconds "
  }, {
    "qosMetric" : "Size of data-in",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "1000",
    "unit" : "KB "
  }, {
    "qosMetric" : "Size of data-out",
    "priority" : "High ",
    "requiredLevel" : " equals to ",
    "value" : "1000",
```

```
94      "unit" : "KB "
95    }, {
96      "qosMetric" : "Link Bandwidth",
97      "priority" : "High ",
98      "requiredLevel" : " equals to ",
99      "value" : "6 ",
100     "unit" : "Kbps "
101   } ],
102   "resourceSpecification" : {
103         "configurationRequirement" : [
104       {
105       "configurationFeature" : "deviceType",
106        "value" : "Gateway "
107       },
108          {
109       "configurationFeature" : "numberOfDevices",
110        "value" : "4 "
111       },
112        {
113       "configurationFeature" : "mobilityOfDevice",
114        "value" : "fixed "
115       },
116        {
117       "configurationFeature" : "communicationMechanismWithIoTDevice
            ",
118        "value" : "push "
119       },
120        {
121       "configurationFeature" : "
            communicationTechnologyWithIoTDevice",
122        "value" : "WiFi "
123       },
124         {
125       "configurationFeature" : "communicationMechanismWithCloud",
126        "value" : "push "
```

```
      },
       {
      "configurationFeature" : "communicationTechnologyWithClould",
       "value" : "WiFi "
      },
      {
      "configurationFeature" : "cpuCapacity",
       "value" : "3 ",
       "unit"  :"GHz"
      },
      {
      "configurationFeature" : "memorySize",
       "value" : "4 ",
       "unit"  :"GB"

      } ]
  }
}, {
   "activityName" : "Analyse small-scale realtime data ",
   "slo" : [ {
    "qosMetric" : "Throughput",
    "priority" : "high ",
    "requiredLevel" : "greater than ",
    "value" : "1000 ",
    "unit" : "Kbps "
  }, {
    "qosMetric" : "Latency",
    "priority" : "high ",
    "requiredLevel" : " less than",
    "value" : "5 ",
    "unit" : "milliseconds "
  } ],
  "resourceSpecification" : {
    "configurationRequirement" : [  {
      "configurationFeature" : "Memory Size",
```

```
      "value" : "4000 ",
      "requiredLevel" : "greater than ",
      "unit" : "KB "
    }, {
      "configurationFeature" : "vCPU Capacity",
      "value" : "2800",
      "unit" : "GHz Xeon "
    },{
      "configurationFeature" : "Hypervisor",
      "value" : "Xen "
   }, {
      "configurationFeature" : "OS Type",
      "value" : "Linux Ubuntu "
   } ]
  }
},{
   "activityName" : "Analyse large-scale realtime data ",
   "slo" : [ {
    "qosMetric" : "Throughput",
    "priority" : "high ",
    "requiredLevel" : "greater than ",
    "value" : "10000 ",
    "unit" : "Kbps "
  }, {
    "qosMetric" : "Latency",
    "priority" : "high ",
    "requiredLevel" : " less than ",
    "value" : "5 ",
    "unit" : "milliseconds "
  } ],
  "resourceSpecification" : {
    "configurationRequirement" : [  {
      "configurationFeature" : "Memory Size",
      "value" : "40000 ",
      "requiredLevel" : "greater than ",
```

```
      "unit" : "KB "
    }, {
      "configurationFeature" : "vCPU Capacity",
      "value" : "44800 ",
      "unit" : "GHz Xeon "
    },{
      "configurationFeature" : "Hypervisor",
      "value" : "Xen "
   }, {
      "configurationFeature" : "OS Type",
      "value" : "Linux Ubuntu "
   } ]
  }
}, {
  "activityName" : "Store structured data ",
  "slo" : [ {
      "qosMetric" : "Availability",
      "priority" : "high ",
      "requiredLevel" : "greater than ",
      "value" : "99 ",
      "unit" : "% "
    } ],
  "resourceSpecification" : {
    "configurationRequirement" : [ {
      "configurationFeature" : "storage Size",
      "value" : "99 ",
      "requiredLevel" : "greater than ",
      "unit" : "TB "
    }, {
      "configurationFeature" : "Storge Type",
      "value" : "SAN-HDD(Storage area network) "
    }, {
      "configurationFeature" : "Geographical location",
      "value" : "USA "
    } ]
```

```
232     }
233 }]
```

Appendix D

SLA for Case study 2 in Chapter 8

```
1
2
3  [  {
4    "appType" : "Intelligent Surveillance",
5    "startDate" : "Wed Nov 21 00:00:00 GMT 2018",
6    "endDate" : "Thu Nov 21 10:35:46 GMT 2019",
7    "sloAtApplicationLevel" : [ {
8      "qosMetric" : "Response Time",
9      "priority" : "High",
10     "requiredLevel" : "less than",
11     "value" : "1000",
12     "unit" : "milliseconds"
13   }, {
14     "qosMetric" : "Cost/Price",
15     "priority" : "High",
16     "requiredLevel" : "less than",
17     "value" : "1000.0",
18     "unit" : "$ per contract period"
19   }],
20   "slaid" : "Intelligent Surveillance Wed Nov 21 00:00:00 GMT 2018
        Thu Nov 21 10:35:46 GMT 2019d"
21 } ,{
22     "activityName" : "Capture Event of Interest: Motion Detector ",
23     "slo" : [ {
```

```
    "qosMetric" : "Data Freshness",
    "priority" : "High ",
    "requiredLevel" : "late than real time by ",
    "value" : "1 ",
    "unit" : "milliseconds "
  }, {
    "qosMetric" : "Sample Rate",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "3 ",
    "unit" : "kHz "
  } ],
  "resourceSpecification" : {
  "configurationRequirement" : [
      {
      "configurationFeature" : "deviceType",
       "value" : "Sensor "
      },
         {
      "configurationFeature" : "numberOfDevices",
       "value" : "4 "
      },
       {
      "configurationFeature" : "mobilityOfDevice",
       "value" : "fixed "
      },
       {
      "configurationFeature" : "communicationMechanism",
       "value" : "push "
      },
       {
      "configurationFeature" : "communicationTechnology",
       "value" : "WiFi "
      },
      {
```

```
      "configurationFeature" : "cpuCapacity",
       "value" : "1.6 ",
       "unit"  :"GHz"
      },
      {
      "configurationFeature" : "memorySize",
       "value" : "1 ",
       "unit"  :"GB"

      } ]
  }
}, {
    "activityName" : "Examine Captured EoI: Object Detector",
    "slo" : [ {
    "qosMetric" : "Gateway Throughput",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "2 ",
    "unit" : "Mbps "
  }, {
    "qosMetric" : "Network Latency",
    "priority" : "High ",
    "requiredLevel" : " less than ",
    "value" : "2 ",
    "unit" : "milliseconds "
  }, {
    "qosMetric" : "Size of data-in",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "4 ",
    "unit" : "KB "
  }, {
    "qosMetric" : "Size of data-out",
    "priority" : "High ",
    "requiredLevel" : " equals to ",
```

```
94      "value" : "4 ",
95      "unit" : "KB "
96    }, {
97      "qosMetric" : "Link Bandwidth",
98      "priority" : "High ",
99      "requiredLevel" : " equals to ",
100     "value" : "6 ",
101     "unit" : "Kbps "
102   } ],
103   "resourceSpecification" : {
104      "configurationRequirement" : [
105       {
106       "configurationFeature" : "deviceType",
107        "value" : "Gateway "
108       },
109          {
110       "configurationFeature" : "numberOfDevices",
111        "value" : "4 "
112       },
113        {
114       "configurationFeature" : "mobilityOfDevice",
115        "value" : "fixed "
116       },
117        {
118       "configurationFeature" : "communicationMechanism",
119        "value" : "push "
120       },
121        {
122       "configurationFeature" : "communicationTechnology",
123        "value" : "WiFi "
124       },
125       {
126       "configurationFeature" : "cpuCapacity",
127        "value" : "3 ",
128        "unit"  :"GHz"
```

```
129       },
130       {
131       "configurationFeature" : "memorySize",
132        "value" : "4 ",
133        "unit"  :"GB"
134
135       } ]
136   }
137 },{
138    "activityName" : "Analyse large-scale realtime data: Object
          Tracker  ",
139    "slo" : [ {
140     "qosMetric" : "Throughput",
141     "priority" : "high ",
142     "requiredLevel" : "greater than ",
143     "value" : "5 ",
144     "unit" : "Kbps "
145   }, {
146     "qosMetric" : "Latency",
147     "priority" : "high ",
148     "requiredLevel" : " less than ",
149     "value" : "5 ",
150     "unit" : "milliseconds "
151   } ],
152   "resourceSpecification" : {
153     "configurationRequirement" : [  {
154       "configurationFeature" : "Memory Size",
155       "value" : "1 ",
156       "requiredLevel" : "greater than ",
157       "unit" : "KB "
158     }, {
159       "configurationFeature" : "vCPU Capacity",
160       "value" : "2 ",
161       "unit" : "GHz Xeon "
162     },{
```

```
      "configurationFeature" : "Hypervisor",
      "value" : "Xen "
    }, {
      "configurationFeature" : "OS Type",
      "value" : "Linux Ubuntu "
    } ]
  }
}, {
  "activityName" : "Actuate: PTZ Control ",
  "slo" : [ {
    "qosMetric" : "Accuracy",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "90 ",
    "unit" : "%"
  } ],
  "resourceSpecification" : {
    "configurationRequirement" : [
      {
      "configurationFeature" : "deviceType",
       "value" : "Sensor "
      },
         {
      "configurationFeature" : "numberOfDevices",
       "value" : "4 "
      },
       {
      "configurationFeature" : "mobilityOfDevice",
       "value" : "fixed "
      },
       {
      "configurationFeature" : "communicationMechanism",
       "value" : "event-driven "
      },
       {
```

```
198       "configurationFeature" : "communicationTechnology",
199        "value" : "WiFi "
200       },
201       {
202       "configurationFeature" : "cpuCapacity",
203        "value" : "1.6 ",
204        "unit"  :"GHz"
205       },
206       {
207       "configurationFeature" : "memorySize",
208        "value" : "1 ",
209        "unit"  :"GB"
210
211       } ]
212   }
213 }]
```

Appendix E

SLA for Case study 3 in Chapter 8

```
[  {
  "appType" : "EEG Beam Tractor Game",
  "startDate" : "Wed Nov 21 00:00:00 GMT 2018",
  "endDate" : "Thu Nov 21 10:35:46 GMT 2019",
  "sloAtApplicationLevel" : [ {
    "qosMetric" : "Response Time",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "1000",
    "unit" : "milliseconds"
  }, {
    "qosMetric" : "Cost/Price",
    "priority" : "High",
    "requiredLevel" : "less than",
    "value" : "1000.0",
    "unit" : "$ per contract period"
  }],
  "slaid" : "EEG Beam Tractor Game Wed Nov 21 00:00:00 GMT 2018Thu
     Nov 21 10:35:46 GMT 2019d"
} ,{
    "activityName" : "Capture Event of Interest (EoI) ",
    "slo" : [ {
    "qosMetric" : "Data Freshness",
    "priority" : "High ",
```

```
24      "requiredLevel" : "late than real time by ",
25      "value" : "1 ",
26      "unit" : "milliseconds "
27    },
28     {
29      "qosMetric" : "Sample Rate",
30      "priority" : "High ",
31      "requiredLevel" : "greater than ",
32      "value" : "100 ",
33      "unit" : "kHz "
34    } ],
35    "resourceSpecification" : {
36    "configurationRequirement" : [
37        {
38        "configurationFeature" : "deviceType",
39         "value" : "Sensors "
40        },
41           {
42        "configurationFeature" : "numberOfDevices",
43         "value" : "6 "
44        },
45         {
46        "configurationFeature" : "mobilityOfDevice",
47         "value" : "fixed "
48        },
49         {
50        "configurationFeature" : "communicationMechanism",
51         "value" : "push "
52        },
53         {
54        "configurationFeature" : "communicationTechnology",
55         "value" : "WiFi "
56        },
57        {
58        "configurationFeature" : "cpuCapacity",
```

```
        "value" : "1.6 ",
        "unit"  :"GHz"
       },
       {
       "configurationFeature" : "memorySize",
        "value" : "1 ",
        "unit"  :"GB"

       } ]
   }
},
{
    "activityName" : "Examine Captured EoI- Client",
    "slo" : [ {
    "qosMetric" : " Throughput",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "1 ",
    "unit" : "Kbps "
  }, {
    "qosMetric" : " Latency",
    "priority" : "High ",
    "requiredLevel" : " less than ",
    "value" : "2 ",
    "unit" : "milliseconds "
  }, {
    "qosMetric" : "Size of data-in",
    "priority" : "High ",
    "requiredLevel" : "greater than ",
    "value" : "2 ",
    "unit" : "KB "
  }, {
    "qosMetric" : "Size of data-out",
    "priority" : "High ",
    "requiredLevel" : " equals to ",
```

```
94      "value" : "4 ",
95      "unit" : "KB "
96    }],
97    "resourceSpecification" : {
98    "configurationRequirement" : [
99        {
100       "configurationFeature" : "deviceType",
101        "value" : "Gateway "
102       },
103          {
104       "configurationFeature" : "numberOfDevices",
105        "value" : "4 "
106       },
107        {
108       "configurationFeature" : "mobilityOfDevice",
109        "value" : "fixed "
110       },
111        {
112       "configurationFeature" : "communicationMechanism",
113        "value" : "push "
114       },
115        {
116       "configurationFeature" : "communicationTechnology",
117        "value" : "WiFi "
118       },
119       {
120       "configurationFeature" : "cpuCapacity",
121        "value" : "3 ",
122        "unit"  :"GHz"
123       },
124       {
125       "configurationFeature" : "memorySize",
126        "value" : "4 ",
127        "unit"  :"GB"
128
```

```
129       } ]
130   }
131 },
132
133 {
134     "activityName" : "Analyse small-scale realtime data-
          Concentration Calculator",
135     "slo" : [ {
136     "qosMetric" : " Throughput",
137     "priority" : "High ",
138     "requiredLevel" : "greater than ",
139     "value" : "1 ",
140     "unit" : "Mbps "
141   }, {
142     "qosMetric" : " Latency",
143     "priority" : "High ",
144     "requiredLevel" : " less than ",
145     "value" : "2 ",
146     "unit" : "milliseconds "
147   }, {
148     "qosMetric" : "Size of data-in",
149     "priority" : "High ",
150     "requiredLevel" : "greater than ",
151     "value" : "4 ",
152     "unit" : "KB "
153   }, {
154     "qosMetric" : "Size of data-out",
155     "priority" : "High ",
156     "requiredLevel" : " equals to ",
157     "value" : "4 ",
158     "unit" : "KB "
159   } ],
160    "resourceSpecification" : {
161     "configurationRequirement" : [  {
162       "configurationFeature" : "Memory Size",
```

```
      "value" : "4 ",
      "requiredLevel" : "greater than ",
      "unit" : "GB "
    }, {
      "configurationFeature" : "vCPU Capacity",
      "value" : "3 ",
      "unit" : "GHz Xeon "
    },{
      "configurationFeature" : "Hypervisor",
      "value" : "Xen "
   }, {
      "configurationFeature" : "OS Type",
      "value" : "Linux Ubuntu "
   } ]
  }
},
{
   "activityName" : "Analyse large-scale realtime data ",
   "slo" : [ {
    "qosMetric" : "Throughput",
    "priority" : "high ",
    "requiredLevel" : "greater than ",
    "value" : "5 ",
    "unit" : "Mbps "
  }, {
    "qosMetric" : "Latency",
    "priority" : "high ",
    "requiredLevel" : " less than ",
    "value" : "5 ",
    "unit" : "milliseconds "
  } ],
  "resourceSpecification" : {
    "configurationRequirement" : [  {
      "configurationFeature" : "Memory Size",
      "value" : "4 ",
```

```
      "requiredLevel" : "greater than ",
      "unit" : "GB "
    }, {
      "configurationFeature" : "vCPU Capacity",
      "value" : "3 ",
      "unit" : "GHz Xeon "
    },{
      "configurationFeature" : "Hypervisor",
      "value" : "Xen "
   }, {
      "configurationFeature" : "OS Type",
      "value" : "Linux Ubuntu "
   } ]
  }
}]
```